Agile and Iterative Requirement Definition

Efficient requirement definition in Agile and Iterative software development

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Bachelor's thesis

Information Technology

Riihimäki, 22.11.2012

Kati Vainola

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Riihimäki
Information Technology
Telecommunications

Author Kati Vainola **Year** 2012

Subject of Bachelor's thesis Agile and Iterative Requirement Definition

ABSTRACT

A high-quality software product has to fulfill a customer's demands from viewpoints of both the content and schedule, and it has to be free from errors. In addition to fulfilling the customer's demands, successful and profitable software product creation uses the available development resources optimally.

Software product creation is started from defining the requirements. Therefore, requirement definition plays a key role for the successful creation of a software product. Another important contributing factor is the used product creation method itself, by enabling efficient usage of development resources and fastening the software's time-to-market.

The commissioner of this thesis was Nokia Siemens Networks (NSN). The main objectives were to analyze and recommend potential enhancements for the currently used requirement definition practices at the NSN mobile voice solution area, and to provide guidelines for the use of the enhancements in the requirement definition phase.

The theoretical basis of this thesis consists of an introduction to the requirement definition principles, and of a study and a comparison of traditional and agile product creation methods. Based on the theory and analysis of the main drivers for the requirement definition enhancements, three enhancement proposals are presented for requirement definition at the NSN mobile voice solution area. To evaluate the proposals in practice, a piloting phase was arranged with real requirement specification cases. As a result of the piloting phase, it was possible to recommend that the proposed enhancements would be taken into use.

Keywords Requirement definition, product creation process, Agile methods

Pages 47 p. + appendices 11 p.

Riihimäki
Tietotekniikka
Tietoliikennetekniikka

Tekijä Kati Vainola **Vuosi** 2012

Työn nimi Ketterän ohjelmistokehityksen vaatimusmäärittely

TIIVISTELMÄ

Korkealaatuisen ohjelmistotuotteen tulee täyttää asiakkaan tarpeet sekä toiminnallisuuden että aikatauluvaatimusten kannalta. Tämän lisäksi sen tulee toimia moitteettomasti. Jotta olisi mahdollista tuottaa asiakkaiden tarpeiden mukaisia ja kannattavia ohjelmistotuotteita, täytyy saatavilla olevia tuotekehitysresursseja pystyä hyödyntämään mahdollisimman tehokkaasti.

Ohjelmistotuotteen valmistus alkaa vaatimusten määrittelystä. Tästä johtuen vaatimusmäärittelyvaihe on tärkeä koko ohjelmistotuotteen kehityksen onnistumisen kannalta. Toinen tärkeä tekijä on käytettävä kehitysmenetelmä, joka vaikuttaa tehokkaaseen resurssien käyttöön sekä aikaan, joka vaaditaan tuotteen saamiseksi markkinoille.

Tämän opinnäytetyön toimeksiantajana oli Nokia Siemens Networks (NSN). Työn päätavoitteina oli analysoida ja suositella mahdollisia kehityskohteita verrattuna nykyisiin vaatimusten määrittelyssä käytössä oleviin menetelmiin NSN mobile voice -osa-alueella, sekä tuottaa ohjeistus näiden uusien menetelmien käyttöönottoa varten.

Työn teoriapohjana olivat työn kannalta tärkeimmät vaatimusmäärittelyn periaatteet. Teoriaosassa myös tutkittiin ja verrattiin perinteisiä tuotekehitysmenetelmiä uudempiin ketteriin tuotekehitysmenetelmiin. Teorian ja muutokseen vaikuttavien tarpeiden analyysin pohjalta esiteltiin kolme kehitysehdotusta. Jotta näistä kehitysehdotuksista saatiin kerättyä kokemuksia myös käytännössä, osana työtä järjestettiin pilotointivaihe, jossa menetelmiä testattiin todellisissa vaatimusmäärittelytapauksissa. Pilotointivaiheen kokemusten perusteella oli mahdollista suositella kehitysehdotusten ottamista käyttöön.

Avainsanat Vaatimus, tuotekehitysprosessi, ketterät tuotekehitysmenetelmät

Sivut 47 s. + liitteet 11 s.

ABBREVIATIONS

2G	Second Generation, GSM
3G	Third Generation, 3rd Generation, UMTS
3GPP	3rd Generation Partnership Project
CSFB	Circuit Switched Fallback
DSDM	Dynamic Systems Development Method
FDD	Feature Driven Development
GSM	Global System for Mobile Communications
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
LA	Location Area
LTE	Long Term Evolution
MOCN	Multi-Operator Core Network
MSC	Mobile Switching Centre
NSN	Nokia Siemens Networks
R&D	Research and Development
UC	Use Case
UMTS	Universal Mobile Telecommunications System
VoIP	Voice over IP
VoLTE	Voice over LTE
XP	Extreme Programming

CONTENTS

1 INTRODUCTION

Requirement definition is in a fundamental role in product creation. That sets the basis for the whole product success, which in turn contributes to the success of the business and to the overall profitability of the company.

To be successful and profitable, a product needs to fulfill the customer demand and requirements for high quality. At the same time, it has to be possible to use the Research and Development (R&D) resources in the optimal manner when building the product. Therefore, the utilized product creation and requirement definition methods need to be carefully chosen, so that high-quality result is gained and any waste of R&D efforts can be avoided.

The constantly ongoing information and telecommunication technology evolution and environment change make the requirement definition especially challenging: creating a high-quality telecommunication software product is like aiming to a moving target. From this background, it is evident, that the chosen product creation model and requirement definition methods need to be very adaptive to changes. In practice, the responsiveness to change and need for delivering software in shorter release cycles requires a change towards more flexible and agile product creation methods. This process is ongoing in Nokia Siemens Networks (NSN) mobile voice solution environment where agile methods are already used in the implementation phase. However, agile product creation methods are not yet efficiently used end-to-end, i.e. from system specification to verification.

In the context of this study, requirement definition methods for software products in telecommunication industry area are considered. As an output, this study targets to propose enhancements for more efficient requirement definition in NSN mobile voice solution environment, as well as to provide guidelines for the new proposed requirement definition methods. The new requirement definition methods shall support successful incremental software delivery with end-to-end agile product creation method.

For building the theory basis in section 2 Requirement definition in product creation, different requirement layers, basic requirement definition models and main product creation modes are described. An overview to traditional and agile product creation methods is given, and the differences of the methods are compared to understand the impacts for the needed change in the requirement definition phase. A detailed analysis of agile methods is not in the scope of this study. In section 3 Agile and Iterative requirement definition, the key drivers for enhancing the requirement definition methods in NSN mobile voice solution environment are explained and three practical improvements to current requirement definition practices are proposed.

The practical part of this study consists of piloting the proposed enhancements and of giving related guidelines. In section 4 Piloting the enhance-

ment proposals, the proposed improvements are tested with real requirement specification cases at NSN mobile voice solution area. The results gained from the pilot requirement specification cases are compared with a set of reference requirement specification cases. The results of the comparison are analyzed, and a recommendation is provided. Additionally, based on the experiences in the pilot cases, guidelines for utilizing the new proposed requirement definition methods are provided for NSN in section 5 Guidelines for Agile and Iterative requirement definition.

In section 6 Summary and conclusions, a short recapture of the study is given and it is estimated how the targets of the study were achieved. Additionally, further development possibilities are identified.

2 REQUIREMENT DEFINITION IN PRODUCT CREATION

When a new product or a new version of an existing product is created, it has to be known what kinds of needs the product should fulfill. This phase, when the desired product characteristics are analyzed, is called as requirement definition. During requirement definition, requirements from practically countless amounts of sources are collected. Many questions need to be answered while the requirements are collected, as such: "What the product should be like to satisfy the customer needs?", "What the product should do to provide sufficient functionality?", "Which criteria the product needs to fill to qualify as a high-quality product?" and many, many more. The answers to these questions finally describe what kind of functionalities, services, characteristics and attributes the product has to fulfill in order to give value for the user of the product.

To meet the user's needs, a proper requirement has to fulfill certain criteria. The main criteria for a good requirement are that it is correct, complete, clear, consistent, verifiable, traceable and feasible (Get It Right the First Time, 2008, p. 9). Other criteria to be considered for good requirements are that they should be modular and standalone with reasonable priority and clear source. However, a requirement should not be a system constraint. Requirements are not only used to define the product, but they are additionally used for validating how well the product and its' functionality covers the original customer need. Therefore, the requirements must be traceable and verifiable.

From this background, it is well justified to say that the requirement definition phase is the key for the whole product creation success. If a failure in the requirement definition phase happens, it directly impacts the end product quality, and as a result reduces the overall customer perception of the product. In the worst case, the missing or faulty requirement may be discovered only when the product has already delivered to the customer. Therefore, it is very important to identify all issues and risks in requirements as early during the product creation as possible, in order to avoid costly rework and corrections.

The above described high level principles for requirement definition are in practice generic for almost all types of products. The scope of this study is in software products of telecommunications industry. In this section, the requirement definition methods and the main product creation modes that are used for telecommunication software product creation are discussed more detailed. This section gives an overview for the reader about requirement collection and definition methods, as well as describes the generically used process modes for software product creation at the principal level.

2.1 Requirement layers and requirement models

The requirement collection and definition happen at several layers, and various models can be created about how to define requirements for a

software product. In this section, the requirement definition layers are explained together with examples of requirement models.

2.1.1 Requirement layers

To profitably implement a new software product, or a new software product feature, there has to be valid business reasoning in place. Business reason is usually evaluated based on business case value calculations. Increased profitability of the product, additional income due to a new developed product feature, reduced operation cost or improved service level are examples of valid business reasons to implement a new product of a feature. Business reason may also be a need to meet regulatory requirements. Quite logically, if business reasoning why to implement the new product or feature does not exist, and there thus is no sponsor for the software project, it is not worthwhile to initiate the project at all.

Therefore, the business requirements set the basis for the whole software project. By analyzing and validating the business requirements, it is possible to find the goals and objectives for the new product or feature. Setting the goals and objectives is often called as scoping. Scoping is very important to enable the software project to concentrate only to functionality that provides the best value both to customer and for the software vendor. Project and requirement scoping typically happens already before the project start or feature implementation decision is made. A good tool for clarifying the scope is to use User Stories, which help to understand who needs the new functionality and why, and what the new functionality should do from the user perspective. User Story model as a tool to define requirements is described more detailed later in this section.

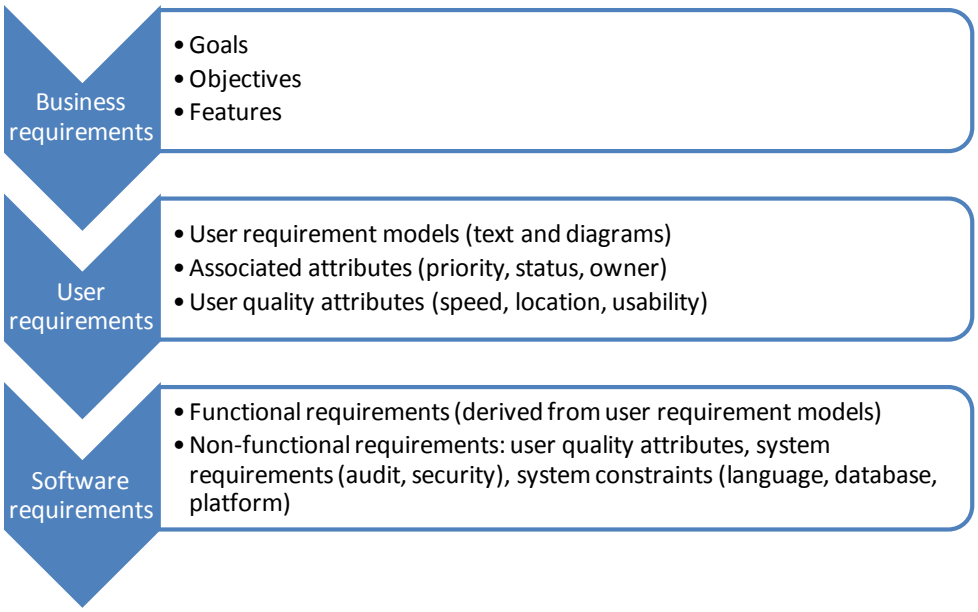

Figure 1 Requirement layers (Gottsdiener, 2002, p. 23)

The actual software requirements cannot be easily derived from the business requirements directly. Therefore, in between of the software requirement layer and business requirement layer, a layer of user requirements (Gottesdiener, 2002) exists as shown in figure 1 Requirement layers.

The purpose of user requirements is to answer to question what the system should do from end user perspective to fulfill the business requirements. This layer of requirements is usually most difficult to define, because it needs to be understood where the business value is coming from to be able to choose a right scope for the new feature or functionality. On the other hand, the technical constraints of the product or system need to be understood. At the user requirement layer, Use Cases can be used to describe what the new functionality should do from the user perspective. Use Case model as a tool to define user requirements is described more detailed later in this section.

The actual software requirements consist of the functional and non-functional requirements, which are based on the user requirements. Functional requirements are practically the actions that the software must take to perform the tasks of the user requirements, and non-functional requirements give the technical constraints for things like capacity and reliability of the product or a new feature (Get It Right the First Time, 2008, p. 39 - 41). When compared to the business and user requirement layer, where questions “what”, “who” and “why” are primarily answered, at software requirement layer the question “how” is additionally considered.

2.1.2 Requirement models

A common language is needed to be able to discuss the requirements among all participants involved in the software project. For communication purposes, different types of requirement models exist. With the structured models, it is easier to share ideas, and take decisions which requirements will finally be needed to implement the product or feature.

Requirement modeling can be done, for example, based on requirement focus or requirement view (Gottsdiener, 2002, p. 28) as shown below in figure 2 Examples of requirement models. Different kinds of models can be chosen for different kinds of needs, instead of using all models for specifying the requirements. In some cases, it may be beneficial to use a combination of different models too.

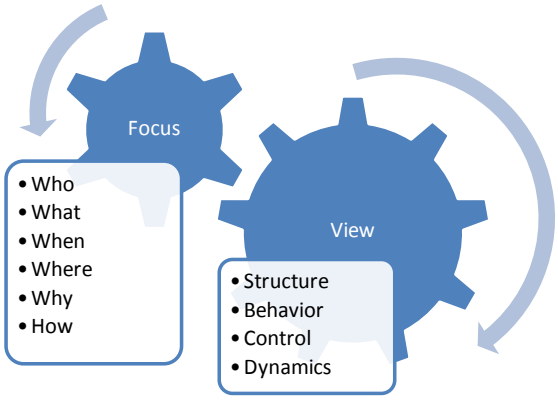

Figure 2 Examples of requirement models (Gottsdiener, 2002, p. 28)

Focus based requirement model concentrates on the six basic question words, which help to estimate quickly the focus of the new feature or functionality. Focus based requirement model can be therefore used, for example, from the beginning of the software project to fast evaluation of overall system impacts and their magnitude. Focus based requirement model helps to reveal which type of additional requirement models, e.g. Use Case descriptions, signaling flows and actor maps, are needed to fully describe the requirements.

For more detailed requirement analysis, various views based on requirement models can be used. Instead of listing all the different possibilities exhaustively in this study, few of the most commonly used and therefore the most important view based requirement models from this study viewpoint are shown. View based requirement models can be divided to behavioural and integrated models. For example, Use Stories and Use Case descriptions represent a behavioural model, i.e. they describe functionality, whereas Domain models represent a structured model, i.e. it gives a static environment view. User Story, Use Case and Domain view requirement models are the most centric from this requirement definition enhancement study viewpoint and they are described more detailed below.

User Story model

User Stories are the most popular way to define the scope for agile and lean software projects (Adzig, 2011, p. 67). User Stories enable the building of a common understanding about what needs to be done, by whom and why.

A User Story consists of three parts (Adzig, 2011, p. 68): *"As a ___ I want ___, in order to ___."* There are also alternative wording possibilities to describe the same items as in the example format above, but the example above is the most typically used format.

By answering to questions “who”, “what” and “why”, each User Story has a clear business value. User Stories do not describe how things are done, which leaves freedom to decide on the implementation details later, once the scoping for the new product or feature has been completed. Therefore, User Stories are an excellent tool to define the business requirements.

When User Stories are defined as a team of participants from both business and development organizations, the created User Stories help to increase the overall business value understanding of the members of the software project.

Use Case model

At the next level of detail, but still in the early phases of the product creation process, Use Cases are typically defined to describe the requirements the user has for the system as well as the interactions between the system and the user. As in the User Stories, it is not yet defined that how the system should operate, instead it is described what the user may do with the system.

Use Case description contains more detailed information about the new functionality than a User Story. A Use Case is an independently verifiable piece of functionality. As shown in Figure 3, it is described what preconditions exist for entering the Use Case, what happens during the Use Case, and what are the post conditions after the Use Case has been executed. There may additionally be sub Use Cases as well as exceptions described.

	Use case title
Purpose	
Actor	
Preconditions	
Description	
Post conditions	
Exceptions	
Performance	
Sub use cases	
Dependent requirements	

Figure 3 Example of Use Case description format

A Use Case description may be complemented with additional requirement models, e.g. a signaling chart or a state chart when relevant.

Domain model

In the contrast to Use Case model, the Domain model view describes the environment via e.g. static product areas.

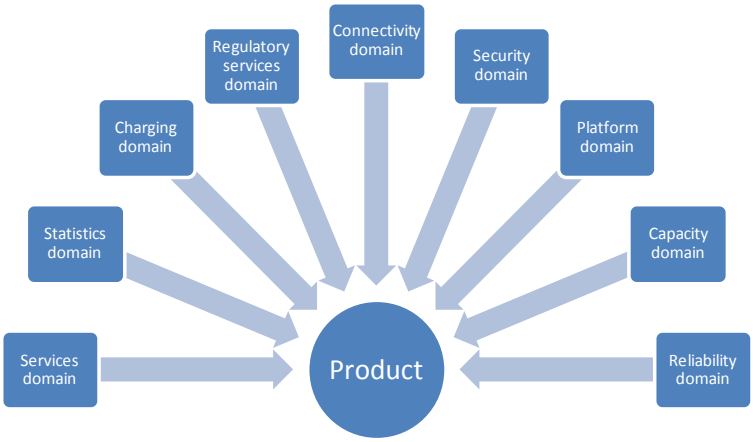

Figure 4 Domain view

The product domains may be the sources for non-functional requirements for the software product (Get It Right the First Time, 2008, p. 39). For example, in the figure 4 Domain view, the capacity, reliability and security are potential non-functional requirement sources for a new feature of the product.

2.2 Product creation models

Product creation process is a description about how products are defined, developed, and managed. A well-defined product creation process enables value creation for customers with optimal R&D usage and helps in building high-quality products. Description of product creation process typically contains e.g. what are the main project milestones and decision points, and what actions and deliverables are needed to successfully create product offering based on the customer needs. Within one company, the used product creation processes may vary between different product areas, and naturally: each company has their own product creation processes to the best suit for the product offering of the company.

A product creation model is a way how the product creation process is carried out. The model is a description of the method. Several creation models exist, which can be utilized for a product development. For example, the newer create models such as Agile and Iterative have been taken into use widely in software industry during 21st centuries, and traditional Waterfall model is still utilized where applicable too. Each creation model has its' benefits and drawbacks. In many cases, especially in larger software product environment one single creation model cannot be applied only. To meet the needs of such products and projects in an optimal manner, a combination of several creation models and practices may be necessary.

For this study, the relevant product creation models are the Waterfall and, Agile and Iterative product creation models. These are described and compared in the following sections. For enabling the comparison, the subsequent main phases of software product creation are used in the example figures of the presented creation modes: requirement definition, software design, implementation, functional testing, system verification and delivery. In the table 1, the meanings of the mentioned product creation phases are clarified in the context of this study.

Table 1 Product creation phases

Product creation phase	**Description**
Requirement definition	Process of collecting the business and user requirements for e.g. software product release to be implemented, answers to question "What".
Software design	Process of collecting the software requirements and designing the software implementation for e.g. product release at more detailed level, answers to questions "What" and "How".
Implementation	Process of implementing the software product based on the requirements for e.g. a product release.
Functional testing	Process of testing the implemented functionality for various software building blocks separately in e.g. a product release.
System verification	Process of integrating the software building blocks and verifying the overall functionality of e.g. a product release.
Delivery	Process of delivering the implemented and verified functionality, e.g. a product release, to the customer.

2.2.1 Traditional product creation models

From this study viewpoint, the most important and at the same time the most well-known conventional product creation model is the Waterfall model. Traditional product creation models include also e.g. Spiral Model and Unified Process models.

The traditional models have in common that a defined and documented stable set of requirements is needed at the beginning of a project. Therefore, these are called as heavy-weight methodologies in comparison to the newer light-weight agile methods. (Awad, 2005, p. 3.)

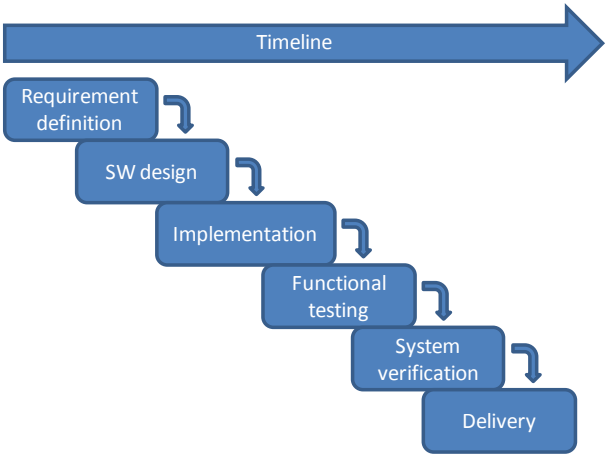

Figure 5 Waterfall model

As shown in figure 5, the different phases of the software project follow each other in a downward shifting manner, and the phases form a pattern like a waterfall. In the basic Waterfall model, the next phase is entered only after the previous phase has been fully completed, but in practical implementations of the Waterfall model, the phases may partially overlap as presented in the figure. Another resemblance to the waterfall is that once a certain checkpoint or project milestone has been reached, there is no possibility to return to an earlier phase.

Despite newer product creation models have widely replaced Waterfall model already, it still has some advantages too. For example, it is a very well-known for software developers and therefore, easy to use (Waterfall 2012). Additionally, the software project is easy to follow and manage because of the used phased approach. Each phase produces documentation as output, so when a new developer is involved with the project, all the produced information about the project is available.

The main disadvantage, on the other hand, is that the Waterfall model is very inflexible regarding additional requirements discovered during the project, after a certain project phase has been reached. If, for example, a new customer requirement is received while the project is already in verification phase, there are no practical means to return to the beginning with the model without a need for rework in all the preceding phases. This naturally is not cost efficient, and it easily creates delay for the overall project schedule. Since the next phase can be entered only after the previous phase has been completed, waste of resources may be experienced while waiting for the previous phase to be ready. (Waterfall 2012.)

In the Waterfall model, testing and verification phases happen quite late in the project. Thus if there is a fault found during e.g. system verification, it takes a long time after development phase until it can be found out and corrected. If serious faults are found, it may ultimately delay the overall project schedule. (Waterfall 2012.)

2.2.2 Agile product creation methods

The main purposes for creating the newer light-weight agile development methods are an ability to respond to constant changes and to bring better value to customer. Compared to the traditional heavy-weight methods, in the agile methods process centrism is reduced by introducing "lightness" in the projects (Awad, 2005, p. 8).

Instead of being one single and simply defined method, agile methods are actually a collection of various iterative development methods, which have in common that development happens in short, time boxed iterations with evolutionary steps. For example, Scrum, Extreme Programming (XP), Feature Driven Development (FDD), Lean, Crystal's methods family and Dynamic Systems Development Method (DSDM) are all agile methods. The agile methods share the same values and principles, but depending on e.g. the technology area, product type and size of the project, these methods may be used differently. There are additionally different variations of agile methods in use in different companies.

In general, agile methods are a very large topic and in the context of this study, it is not beneficial to repeat all the already written information in the various comprehensive books about agile development methods. Instead, the key thinking behind agile methods is shown via introducing the Agile Manifesto. An overview is given to Agile and Iterative development model and its' interaction with requirement definition. Among the several agile methods, Scrum is the most relevant agile method related to this study, and thus it is chosen to be shortly described as an example, in order to illustrate the related practices.

2.2.2.1. Agile Manifesto

Agile Alliance is a non-profit organization with global membership, which is committed to advancing agile development principles and practices. The Agile Alliance has defined the "Agile Manifesto" (Agile Alliance 2012), with following main principles:

- Individuals and interactions over processes and tools
- Working Software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following the plan

Even though according to Agile Manifesto importance is seen also for the items on right side, the items on left side are more valued. In a nutshell, agile methods value communication and collaboration in order to provide software that meets the customer need.

2.2.2.2. Agile and Iterative development models

Iterative development means growing the product in small steps in series of time-boxed iterations, aiming for a release which is externally released

to customer. In figure 6 Iterative and incremental development, these principles are shown in more detail.

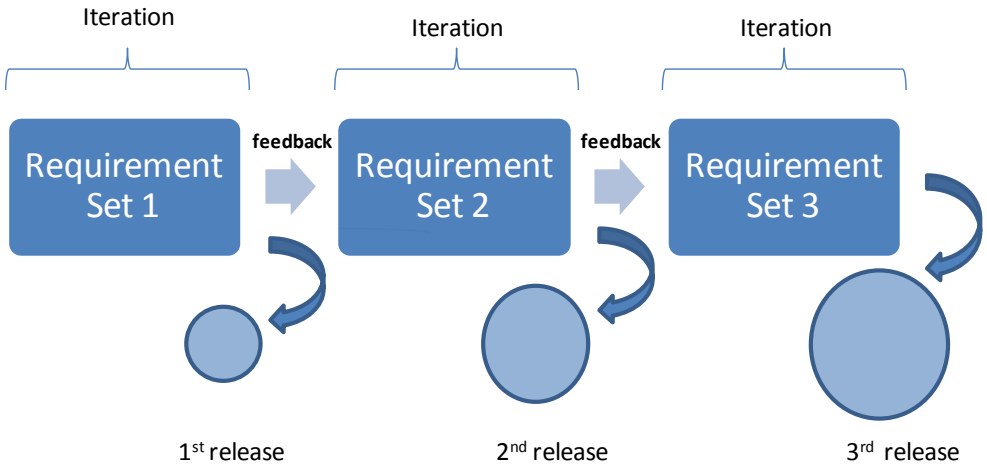

Figure 6 Iterative and incremental development (Larman, 2004, p.10)

A time-boxed iteration is typically a period which lasts 1 – 6 weeks, after which working, integrated and tested software is released for the agreed content of the iteration (Larman, 2004, p. 267). One iteration contains requirement definition, software design, implementation and verification for the requirement set of the iteration. Thus, the iteration is a mini-project by itself, within the larger software product project. Time-boxed iteration means that if all requirements can't be developed by the iteration target date, requirements should be removed from the iteration instead of moving the iteration deadline.

One of the pitfalls of Agile and Iterative process is that unless the process is incremental in addition that it is iterative; it easily becomes just a series of mini-waterfall projects. Therefore, in the agile and iterative process, it is important to learn continuously from the earlier iterations as well as incorporate new requirements, to successfully increment the implementation towards the final solution.

The ultimate target of iteration in theory is that a working product or a new feature could be released to customers after each iteration. However, in larger-scale software products this is usually not feasible, and therefore, most of the iterations result to internal software releases. It is possible to group the iterations to form a meaningful internal release from the functionality viewpoint. Despite being an internal release, all intermediate releases should be fully integrated and tested, but they contain only partial functionality.

In practice in the first iterations of a software project, parallel to early software development, relatively more effort is given to requirement definition. The intent is to discover the architectural constraints and to find high customer value or risky Use Cases and requirements (Larman, 2004, p. 319). Unlike in traditional development methods, feedback from earlier

iterations is available, based on which it is possible to find additional requirements and make necessary adjustments to the content of upcoming iterations.

Especially in large software projects, a significant amount of changes in requirements can be expected: *"Even medium-sized projects have change rates around 25%; on very large projects, it is 35% or more"* (Larman, 2004, p. 51). Larman bases this statement on studies made by Jones, Boehm and Papaccio on a large amount of software projects. Therefore, it is of utmost importance, that used software-development method is adaptive to requirement changes.

Agile methods have their downfalls too. Utilizing agile methods in global and large development projects is more challenging than using the traditional product creation methods. Without predefined project scope and schedule, the program costs may be more difficult to estimate. Implementing high-quality software requires strong involvement of business representatives who may be sometimes difficult to engage in the project, unless there is a strong customer demand in place. (Awad, 2005, p. 8.)

2.2.2.3. Scrum

In Scrum method, self-directed teams are in centric position. The scrum team commits to the defined iteration goal, and it is given the authority to make decisions on how the goal can be met. To enable focus in achieving the goal of the iteration, scrum team should not receive additional tasks during the iteration. Sufficient resourcing and removal of potential obstacles that would prevent to achieving the target are at the responsibility of the management and scrum master.

A scrum master is one of the members of the scrum team, with additional responsibility to interact between the project management and the team, as well as to run the scrum meetings. Scrum master also gives a demo at the end of the iteration to external stakeholders to show the functionality implemented within the iteration.

Another important role in the Scrum method is the role of Product Owner. Product Owner represents the customer view and is in responsibility of prioritizing the functionalities that will be implemented within the iterations. Product Owner additionally participates in reviewing the system at the end of the iteration and chooses the targets for the next iteration. Because Scrum is a customer driven method, for successful operation it is required that the Product Owner is involved in the process and is able to make decisions on which items are at high priority.

The functionality prioritization typically is done via Product Backlog, which a list of implementable items with their given priorities and status information, among other relevant information. Product Backlog is openly available and thus the items and priorities are visible for all participants of the Scrum teams and the whole software project.

Scrum lifecycle consists of four phases: planning, staging, development and release (Larman, 2004, p. 113). Planning and staging form so called pre-game phase, where the funding and content is clarified, including requirement analysis and prioritization for the first iteration. Development phase consists of series of typically 30-day long sprints, during which the actual implementation is done. Release phase targets to deployment of the implemented functionality, including preparation of documentation and conducting necessary training.

The main strengths of Scrum method reside in the self-organized teams with strong focus and adaptability to change, as well as the simple management practices.

2.2.3 Agile and traditional product creation methods in comparison

The main characteristics and differences between Agile and traditional heavy-weight methods are shown in Table 2 Difference in Agile and heavy-weight methodologies.

Table 2 Difference in Agile and heavy-weight methodologies (Awad, 2005, p. 35)

	Agile methods	**Heavy-weight methods**
Approach	Adaptive	Predictive
Success measurement	Business Value	Conformation to plan
Project size	Small	Large
Management style	Decentralized	Autocratic
Perspective to change	Change Adaptability	Change Sustainability
Culture	Leadership-Collaboration	Command-Control
Documentation	Low	Heavy
Emphasis	People-Oriented	Process-Oriented
Cycles	Numerous	Limited
Domain	Unpredictable/Exploratory	Predictable
Upfront Planning	Minimal	Comprehensive
Return on Investment	Early in Project	End of Project
Team Size	Small/Creative	Large

As described earlier, in traditional Waterfall type of development method, the different product creation phases follow each other. In Agile and Iterative product creation method, each of the iterations contains the same product creation phases, but only for a partial content of the software project. The difference between traditional and agile methods is illustrated in figure 7.

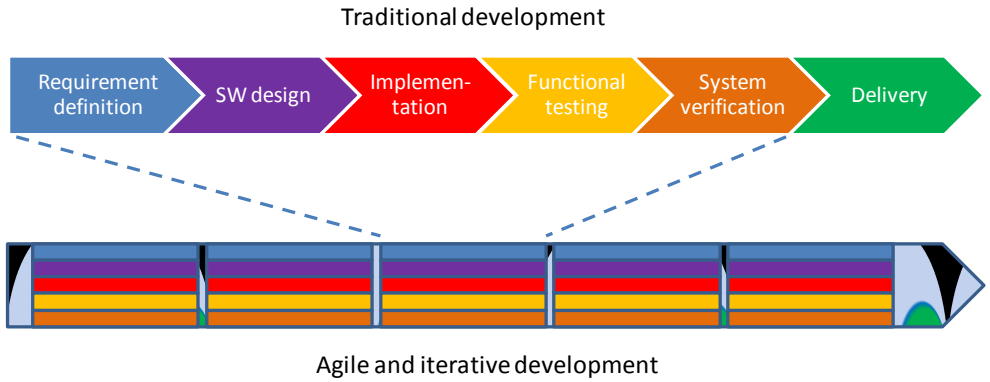

Figure 7 Traditional product creation vs. Agile and Iterative product creation methods

For Agile and Iterative development, there are several key motivations (Larman, 2004, p. 51 - 53):

- Lower risk via early risk discovery and mitigation
- Easier management of requirement complexity and change
- Early partial product availability
- Higher quality with better match to customer need
- Increased communication

Agile and Iterative methods differ from traditional methods, especially on how risks, complexity and uncertainty are managed. Where traditional methods aim to create reliable work effort and time schedule estimates via careful before-hand planning and analysis, in Agile and Iterative methods, the estimates are based on collecting experiences about implementation of the most critical parts of the system and features. In Agile and Iterative development, risks can be identified earlier, because integration of the different parts of the system is done at the end of each time-boxed iteration. In traditional models, the integration is done only after the whole project content has been implemented, and therefore any emerging unexpected issue may lead to rework and potentially even cause delay for the project.

In Agile and Iterative development, every iteration is started by planning the content of the iteration. At this phase, it is possible to evaluate the included requirements against the customer need and take into account the findings and learning from the previous iterations. Therefore, it is much easier to adopt to change, as well as manage the overall complexity both from project management and implementation viewpoints, because only a subset of the whole functionality of the release is implemented within the iteration. The lower complexity level results into higher quality of the produced software.

In traditional development method, requirement definition is normally done for the full project content before entering the software design phase. In Agile and Iterative development on the other hand, requirement definition is done only for such a feature and functionality, which is planned to be implemented and verified within the same iteration. The benefits of this approach are that it is possible to prioritize the features and functionalities

to have the best possible match with the customer demand both from content and timing point of view. This way, it is possible to concentrate in an iteration only to the prioritized features and functionalities. When it is possible to implement only the highest priority functionalities also the waste of effort is reduced: everything that is specified for iteration will be implemented and verified. In the traditional model, when the customer's need changes during the project, it is possible that work for some unnecessary functionality is either partially carried out, or in worst case functionalities that will not be ever needed by a customer get implemented.

In some cases, it is beneficial that a product demo or trial system will be provided for the customer. With iterative development, this possibility is better inbuilt to the development model when it is possible to choose the content of the early iterations to match the customer demo and trial system needs.

Traditional development model relies to documentation and training as a main source of information and competence transfer between the different product creation phases. In agile models, there is stronger engagement of the team members by frequent interaction, and working together in cross-functional teams reduces the need of formal information delivery from one person to another. This, however, does not mean that documentation or training would not be needed in agile model at all; it is still necessary to document, for example, the requirements for tracing possibility and for verification needs.

In addition to the many benefits shown above, it is even possible to put a monetary price tag for iterative development. This can be calculated based on factors like increase of productivity, increased quality of the product and the project, reduced failure cost and fit of the product to the true customer demand (Larman, 2004, p. 100 - 102),

Overall, it can be summarized that compared to the Agile and Iterative models; the traditional Waterfall development model is not anymore flexible enough in the modern software projects. Meeting the customer demand, time to market, responsiveness to changes and cost efficiency are in the key roles of successful software development.

3 AGILE AND ITERATIVE REQUIREMENT DEFINITION

There are several constraints, which contribute to software-development success in general. The delivered software has to provide the right solution for the exact customer need at a right time, with high quality. High quality in this context means error free software, which matches the customer need. From the software vendor viewpoint, the customer demand needs to be filled with optimal usage of R&D resources, and therefore planning the software release schedule and release cycle to make the required software available at the right time is important. Especially in large software projects these are real challenges, which require careful balancing. These main challenges are summarized in Figure 8.

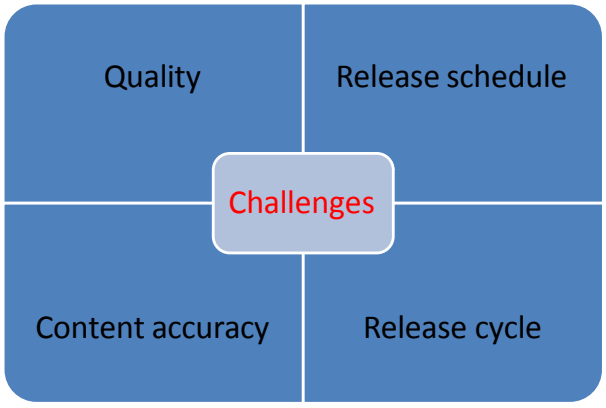

Figure 8 High level challenges of software development

These challenges together with the fact, that requirement definition is one of the key areas for successful software delivery, have brought the need to study possible enhancements compared to the current way of defining the requirements in NSN mobile voice solution area.

As background information for the study, the NSN mobile voice solution environment where the study is carried out is described. It is also explained what are the targets and expectations for the Agile and Iterative requirement definition. Based on the targets and expectations, a set of proposals for more efficient requirement definition is given at the end of this section.

3.1 Description of NSN mobile voice solution environment

The study of requirement definition enhancement is carried out for Nokia Siemens Networks (NSN), which is a world's specialist company for mobile broadband solutions. More specifically, the study concentrates to NSN's Mobile Switching Centre (MSC) Server System solution area, where NSN is the leader in mobile voice solutions. MSC Server System offers support for e.g. 2G/3G, Voice over IP (VoIP) and Long Term Evolution (LTE) voice solutions. Product –wise the MSC Server System con-

sists of MSC Server product for control plane signaling management and Media Gateway product for user plane subscriber data payload management.

From software viewpoint, MSC Server builds on a strong legacy of MSC network element. The NSN MSC was first developed for GSM networks in early 1990s for handling signaling at control plane and managing voice payload at the user plane for mobile voice services. MSC has evolved since into a multi-access capable soft switch server product with thousands of added features, consisting of approximately 20 million lines of software code, and is currently running in several hundred operator's live networks.

One of the main architectural changes was done in beginning of 2000-decade, when the user plane management functionalities were separated from MSC into a standalone Media Gateway network element, to enable the bearer-independent circuit-switched core network. As result of this control plane and user plane handling separation, MSC Server and Media Gateway products together form a system solution for mobile voice services. Software development work is constantly ongoing for MSC Server System because of continuous evolution towards future network architecture needs like Voice over IP and Long Term Evolution and to fulfill customer feature requirements. In addition, new product platforms are taken into use for increasing efficiency.

The product creation model for development of such telecommunication software products with a long history has its' legacy as well; even though the software implementation phase (including software design, implementation and functional testing) are already carried out in Agile and Iterative model for MSC Server System. The system-level functionalities like specification and network verification are still in large extent following the traditional Waterfall product creation model. To better support the agile software implementation and to overcome the negative impacts of the mixed working models of development, a change in both the system specification and network verification working models towards Agile model is needed. From this viewpoint too, the system specification especially should better enable new features and functionalities to be implemented and verified in smaller entities, which would fit into the iterations of one release.

Organization –wise, there are challenges as well: the development of MSC Server system has been divided to several countries on two continents. Therefore, good and frequent communication between all parties involved in the software project is required. Working in virtual teams in matrix organizations across several time zones is a normal operation model in the MSC Server system projects.

3.2 Targets and expectations for Agile and Iterative requirement definition

The main goals for requirement definition enhancement are:

- Improved software quality and faster release cycle
- Improved requirement setting accuracy and requirement quality

- Increased competence development and information sharing

These main goals are explained more detailed in the following subsections.

3.2.1 Improved software quality and faster release cycle

High quality of software products is one of the basic competitive requirements. On the other hand, lack of quality very soon will impact the overall success of the software product by negatively impacting the customer perception of the product and the whole software supplier company. Therefore, it is of utmost importance, that any quality issues and faults in the software are identified and corrected as soon as possible.

There already are strict quality criteria in place in NSN product creation to ensure that only high-quality products that fulfill the defined criteria, are delivered to the customers. Software quality is ensured in all phases of the product creation process, e.g. by reviews of the product artifacts like specifications and by verification of the produced software at different levels. However, the phase when the errors in the software which have the widest impact at system level are found, is often quite late in the project, e.g. in system verification when the different products with new software are finally put together as a system. Correction and reworking usually takes some time, which delays the final verification and acceptance of the software. Therefore, in the worst case, the whole software release may be delayed.

The preventive action against project delay is to perform system-level verification earlier in the project than currently. It means a change towards Agile working model in the system verification phase too. This correspondingly sets pressure for system specification and requirement definition: the most important and riskiest system-level Use Cases and requirements need to be available early enough in the project. This is necessary to enable software release and iteration planning, and to enable planning and running the necessary test cases already during the iterations of the project. Based on the requirements and Use Cases, it has to be possible in verification planning phase to reach sufficient test coverage for the functionalities included in the iterations with an optimal test case amount.

When missing requirements are identified during iterations, they can be implemented and verified during the coming iterations of the project. Software development becomes more efficient, because less visiting in the previous phases has to be done, and thus less amount of rework is needed when compared to Waterfall model.

From a project management viewpoint, this approach brings more visibility to the actual software project readiness. When a better visibility and predictability to the software project is gained, the project planning becomes easier and project delays can be to a greater degree avoided, which in turn helps to shorten the release cycle.

3.2.2 Improved requirement setting accuracy and requirement quality

From customer satisfaction perspective, there has to be a right focus in each software release to match the customer need. Finding the right focus requires, that there are sufficient business and technical understanding about the customer needs in place when the release or feature scope is being defined.

In the current working practice at NSN, the Use Case and requirement definition for a new feature is typically done mainly by a feature expert. The feature expert carries out the feature requirement specification work by consulting other relevant experts about relating business and technical aspects. Regardless of arranged work meetings during the requirement definition process and finally reviewing the ready work product, in this working method it is possible that all different viewpoints and interworking issues are not identified. Even the feature scope may partially miss the customer demand, if the scope is not correctly understood. Later, additional specification work is needed to cover the missed Use Cases and requirements.

It can be argued that the main reason for this is not lack of competence, but instead it is more about lack of communication. One person, despite how experienced, cannot be an expert in every aspect from the feature business value analysis to delivery. Especially the product environment with plenty of interworking features and functionalities increase the complexity level even further. The overall required experience and competence, however, reside in the organization, but it is distributed to several organizational areas and experienced individuals. Therefore, more cross-functional team effort and communication are needed for feature scoping. Equally, team effort brings benefits for clarifying and choosing the right Use Cases for a software release from business and technical viewpoints.

If the real customer needs and business requirements are not clearly understood, it may have an impact on testability of the new functionality as well. In the worst case, it may happen that during testing the emphasis is in verifying that the implemented software is just according to what has been defined in the earlier system specification and implementation phases, while some of the actual customer requirements may be neglected. On the other hand, if the feature scoping and Use Case selection has been done successfully to match the customer need, it is possible to define an optimal set of acceptance test cases, which are used to ensure that the customer need is fulfilled.

A successful Feature scoping and Use Case selection, which matches to the customer real need, makes possible to divide the features into smaller entities. For example, instead of specifying a full-blown system-level feature, which requires a year to implement, it may be better solution from the customer viewpoint to have a trial solution available for a subset of feature functionalities within a couple of months.

3.2.3 Increased competence development and information sharing

As mentioned earlier, a huge amount of competence resides in the organization, but full benefit of competence of the experienced individuals may not be cashed out in the current way of working. On the other hand, additional team work in the requirement definition phase would enable to share the knowledge between all related parties right from the beginning, to create a common understanding on the feature under development. When the new feature is handled from different angles from business aspects to verification, wider understanding of the problem area and requirements is gained. This helps to ensure that the most important Use Cases and requirements will be successfully managed through the product creation process for the release.

When the experts from different areas are collected into virtual groups to work with the new feature or functionality, everyone can support the group by bringing in his or her own competence. This way, the virtual group, which may also be called as a feature team, should be able to deliver high-value results much better than an individual can do alone.

In the traditional product creation process, the shift between the different phases is very much dependent on produced documentation, for example, requirement specification documents. In additional, formal training, events are organized for competence sharing to the next product creation process phases after the system specification has been completed. Even though these events naturally, to some extent, are increasing the overall understanding about the release scope and content, they do not yet give all the necessary information for the following phases to start their work efficiently. When the following phases are actually starting their work for the release, additional clarification sessions and planning meetings are needed about the new feature or functionality. The need for such additional sessions can be reduced by forming feature teams at the beginning. This allows participants to learn what the feature or new functionality is all about, with a possibility to add their own contribution to make the Use Cases and requirements more meaningful to perform the tasks, that are needed in the other phases of product creation too.

The overall teamwork benefits in innovative projects have been empirically studied, for example, by Hoegl and Gemuenden (2001). It has been shown that teamwork positively impacts into project success by increased quality, as well as by increased efficiency of project schedule and budget (Hoegl and Gemuenden, 2001, p. 439).

The six teamwork quality construct factors are (Hoegl and Gemuenden, 2001, p. 437 - 438) are:

- Communication: Information sharing and information flow within and between the teams is a direct prerequisite of project success.
- Coordination: Team is able to agree on a common target and tasks, and divides the work into smaller subtasks. Team members are able to carry out the subtasks independently and efficiently, without overlap between the tasks of other team members.

- Balance of member contributions: Contributions to the team tasks is balanced based on the expertise and experience of each individual team member. Everyone is able to bring in the relevant knowhow into the team work.
- Mutual support: Team members are able to assist each other, and further develop ideas towards the common goal.
- Effort: The required effort to carry out the tasks can, and should be, fairly shared within the team to share the workload.
- Cohesion: When the team members feel that they belong to the team with a common goal, it maintains the motivation on the team tasks.

Therefore, in addition that increase of teamwork is expected to bring increased competence development and information sharing, it also contributes to the software quality and project efficiency overall.

3.3 Enhancement proposals for Agile and Iterative requirement definition

When considering the above-described goals for requirement definition enhancement, increased communication and competence sharing are in very important roles. In the current technically complex environment, where the competence has scattered to various organizations and continents, additional team effort is clearly required to break the silos between organizations and expertise areas. Without efficient team effort, there is a high risk to miss the core set of requirements for the new feature or functionality.

It is important to use the available R&D resources optimally and meet the project goals and milestones. Development efforts need to be concentrated only to the most important feature or functionality enhancements, via which it meets best the customer demand in both implemented feature content and timing of the software. Any waste in the process needs to be avoided or at least be minimized.

Agile and Iterative method for requirement definition work needs more and regular communication between experts, and it can increase of requirement definition efficiency. These are expected to be achieved by creating virtual feature teams and organizing Use Case workshops for Use Case scoping.

3.3.1 Proposal 1: Feature team

The functionalities of MSC Server system are generically divided into functional domains. In addition to this, new features and functionalities typically belong to a larger feature area umbrella, for which several experts from various teams and organizational functions provide their effort and competence both from technical and business viewpoints.

This approach could be made more concrete by officially forming feature teams around the new feature or functionality. The feature team concept

itself is not a new one; it has been in use in the past already with good results. However, some erosion in this practice has happened in past years, and thus the feature teams could be reinvigorated.

A feature team would consist of representatives from all the different phases of product creation process, and it should work closely together for the whole duration of the feature development process. The feature team would carry the responsibility for the feature area, not only for the duration of the product creation process for the new feature, but also later in the maintenance phase. In practice, feature teams would be virtual; the participants of the feature team should typically come from cross-organizational units, but the line management responsibilities of the feature team participants would still remain in their home organizations. The feature team roles and responsibilities are shown in Table 3.

Table 3 Expertise areas and roles of feature team

Expertise areas	Roles	Expertise and responsible
Project management	Project managers	The persons manage the R&D and verification project resourcing and timing, for example iteration planning.
Feature definition	Feature expert	The person has the overall responsibility and expertise on the feature area.
Requirement specification	Requirement expert	The person is the author of the requirement specification of the new functionality or feature.
Business needs	Business manager or business owner	The person is responsible of the product area and feature business view and business requirements.
Customer requirements	Customer representative	The person has originally initiated the customer request, typically a person from the customer team or a customer representative.
Software design	Software designer	The persons have the responsibility to design and develop needed software.
Implementation	Implementation expert	The persons have the responsibility to carry out implementation specification.
Verification	Testing and verification expert	The persons have the responsibility of the verification of the feature.

Additionally, there may be other expertise required in the feature team, e.g. from platforms, interworking features, network planning aspects and related counterpart product viewpoints depending on the feature in question.

There is some resemblance in the working method of a virtual feature team when compared to the working method of a scrum team: both should be quite self-directed. In scrum teams, the responsibility to co-operate be-

tween the project management and the team is with the scrum master, in feature teams this role is with the feature responsible. Scrum master runs the scrum meetings, and correspondingly the feature responsible should run and facilitate the feature team meetings. On the other hand, the scope of a feature team is larger. When a scrum team concentrates to functionality produced in iteration, a feature team should exist through several product releases as long as needed from the feature viewpoint.

Tasks of a feature team could include for example:

- Participate to feature scoping and solution definition in requirement definition phase.
- Provide support for problem solving during feature development and verification.
- Provide support in problem and fault situations after the feature delivery to the customer.

During especially feature development and verification phases, it would be a good practice to arrange regular feature team meetings for efficient communication, problem solving and coordination purposes.

3.3.2 Proposal 2: Use Case scoping for a release or iteration

When a requirement specification phase for a new feature or functionality is entered, it needs to be decided, which are the main Use Cases for the new functionality to be included in the planned release or iteration. A certain level of feature scoping can be expected to happen already before the requirement specification start decision is made, but in the beginning of requirement specification phase, there typically still is plenty of room for making the content to accurately match the actual customer need. At the same time, the optimal use of the R&D resources in the project need to be ensured. On the other hand, the less important Use Cases and functionalities can be safely left for later iterations of the software project or even to future product releases. Therefore, there is a need to prioritize and group the Use Cases of a feature, to enable them to be handled in correct product release and iteration. This way, the best match with both the customer need and implementation possibilities is achieved.

When scoping and prioritizing the use cases, following topics need to be considered to find the high priority Use Cases for a release or iteration. For example:

- Customer situation: Which Use Cases are the most relevant from customer viewpoint, including both content and timing?
- Technical relevance: Which Use Cases are the main building blocks that need to be available first and which are the riskiest Use Cases that need to be verified first?
- Implementation and verification capabilities: Which Use Cases can be implemented and verified first with the available resources?

For Agile and Iterative product creation, the chosen Use Case implementation and verification should preferably be possible within same sprint.

The Use Cases for a feature or functionality may be grouped to form an entity that can be managed within one iteration or software release. These Use Case groups together build up the Use Case package, which contains the full feature functionality. For very complex features, several Use Case packages may exist, too.

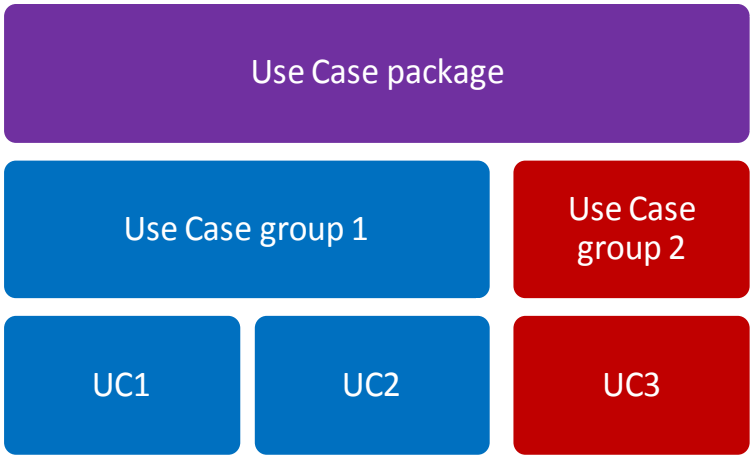

Figure 9 Hierarchy of Use Case, Use Case group and Use Case package

In the figure 9, the Use Case packages and Use Case groups are administrative entities, whereas the Use Cases are the descriptions of an end-to-end functionality. Use Cases can be independently verified and each Use Case should have their corresponding acceptance test cases. With acceptance test cases is it is possible to demonstrate for the customer that the added functionality works correctly and matches the customer need.

In general, the approach to choose the highest priority Use Cases for first release or iteration supports well the targets of Agile and Iterative requirement definition. The main potential pitfall is related to capability of identifying already from beginning the overall technical solutions: same architectural solutions that are chosen for the high priority Use Cases should be usable also for lower priority Use Cases, which are planned to be implemented later. If the overall target functionality of the feature is not correctly foreseen, some changes to already implemented solutions may be required. In similar manner, the legacy software code implemented in the earlier release, or even some earlier made architectural solutions may complicate the Use Case selection in practice.

3.3.3 Proposal 3: Use Case workshop

To successfully perform the Use Case validation and grouping, cross-functional joint effort is needed. Because it is challenging to collect all the relevant experts from various organizations separately, the working model itself should enable arranging Use Case workshops, which would enable the participants to allocate time for the Use Case definition work. In this

aspect, the Use Case workshop participation during the feature scoping and requirement definition phase would most naturally fall to the responsibility of the feature team and its' participants.

The following benefits are expected, when a Use Case workshop is arranged for a new feature or functionality:

- Higher requirement quality and efficiency: Workshop with experienced participants from all different phases of product creation process reduces a risk of misunderstanding the customer need or missing some of the key Use Cases.
- Competence sharing: Common understanding about the feature or functionality, as well as the required Use Cases and their priorities is gained from the beginning.
- Easier and faster work shift between product creation process phases: All parties are able to learn about the new feature or functionality from the beginning, and this way it is possible to enter sooner to the other development phases in parallel, even before the requirement definition phase is not fully completed yet.

The main expected output from a Use Case workshop is the prioritization of the Use Cases as illustrated in figure 10.

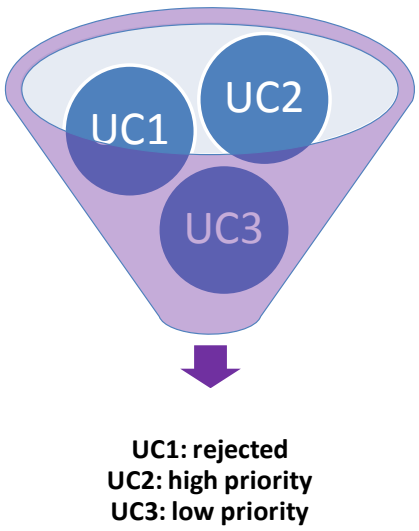

Figure 10 Use Case processing during Use Case workshop

Prioritized list of Use Cases will be the basis of forthcoming specification, implementation and verification activities for the feature or functionality included in the software release. An example of Use Case list, and the information that it should include, is given later in this study in Figure 14 Example of a Use Case list.

As non-tangible but high value output from the Use Case definition workshop, a common understanding of the meaning and level of importance of

each Use Case is established between all the participants. This will help in all future communication related to the new functionality.

4 PILOTING THE ENHANCEMENT PROPOSALS

In the previous section, the targets and expectations as well as the enhancement proposals for Agile and Iterative requirement definition were identified. Due to the nature of the enhancement proposals, they are expected to contribute to achieve more than one of the targets, as shown in Table 4 Target and enhancement proposal mapping below.

Table 4 Target and enhancement proposal mapping

Target	**Enhancement proposal**
Improved software quality and faster release cycle	Use Case scoping
Improved requirement setting accuracy and requirement quality	Feature team, Use Case scoping, Use Case workshop
Increased competence development and information sharing	Feature team, Use Case workshop

In order to evaluate the feasibility to take the enhancements proposals into wider use in the requirement specification phase at NSN mobile voice solution area, a piloting phase for the enhancements is required to collect feedback for the decision making. In addition, guidelines for using the new practices are needed.

Therefore, the practical part of this study consists of two tasks:

- Arranging a piloting phase to collect experiences in real requirement specification cases about using the new proposed methods for feature teams, Use Case scoping and Use Case workshop. The piloting phase is further described in section 4 Piloting the enhancement proposals.
- Providing guidelines for the Use Case workshops and Use Case screening. The guidelines are included in section 5 Guidelines for Agile and Iterative requirement definition.

Section 4.1 Introduction of the chosen pilot and reference requirement specification cases gives a summary about the chosen pilot and reference requirement specifications.

The feedback collection method for the pilot and reference cases, as well as the corresponding evaluation criteria are defined in section 4.2 Piloting feedback collection method.

In section 4.3 Piloting result analysis the experiences are summarized and analyzed. The detailed feedback from each pilot and reference requirement specification is presented in Appendix 1.

Section 4.4 Analysis of the used method contains an analysis of the used comparison method itself. The recommendation is included in section 4.5 Recommendation, which guides the decision making of including the en-

hancement proposals in the NSN's MSC Server System requirement definition process.

4.1 Introduction of the pilot and reference requirement specification cases

The enhancement proposals for Agile and Iterative requirement definition were piloted and experiences were collected via two MSC Server system level pilot requirement specification cases. Additionally, two requirement specifications that have been done already earlier according to the old methods were chosen for comparison.

Case 1: Multi-Operator Core Network (MOCN) support in MSS

Case 1 is a reference requirement specification which specifies the Use Cases and requirements for a standard 3G Partnership Project (3GPP) network sharing solution for NSN MSC Server system.

Because the scope of Case 1 is in a standard based network sharing solution and the amount of involved network elements and interfaces is relatively low, the complexity level to define the Use Cases and requirements for this requirement specification can be considered as medium.

Case 2: IPv4/IPv6 dual stack in MSS system

Case 2 is a reference requirement specification which introduces commercial level IPv4/IPv6 interworking capability for the NSN MSC Server system.

Case 2 is a high complexity requirement specification because of the amount of related external interfaces from the MSC Server system and correspondingly involved co-products. Additionally, IPv4/IPv6 interworking is required at several layers of the involved interfaces.

Case 3: MSS Pool – enhanced control for LTE/CSFB and LA off-loading

Case 3 is the first MSC Server system level pilot requirement specification, which defines additional functionality to already existing MSS pooling feature at network resiliency area.

Even though the amount of supported external interfaces is increased and additional feature interworking requirements are introduced, this requirement specification is of medium complexity level because the base MSS pooling feature environment already exists.

Case 4: Georedundancy solution for VoLTE

Case 4 is the second pilot requirement specification. The purpose of this requirement specification is to collect together in Use Case format the current Voice over Long Term Evolution (VoLTE) solution related geo-redundancy capabilities at system level to enable end-to-end system verification. Due to this, no new actual requirements are introduced in this requirement specification.

This is a high complexity level requirement specification, which covers many MSC Server system external interfaces and several products owned by different product lines for the full NSN VoLTE solution.

The included pilot and reference requirement specification cases are summarized in the table 5 Summary of the reference and pilot requirement specification cases.

Table 5 Summary of the reference and pilot requirement specification cases

Case	**Case type**	**Complexity**	**Amount of included**		
			Interfaces	**Use Cases (active/total)**	**Requirements (active/total)**
Case 1: Multi-Operator Core Network (MOCN) support in MSS	Reference	Medium	2	11/15	14/15
Case 2: IPv4/IPv6 dual stack in MSS system	Reference	High	8	12/12	34/34
Case 3: MSS Pool – enhanced control for LTE/CSFB and LA off-loading	Pilot	Medium	4	11/25	20/20
Case 4: Georedundancy solution for VoLTE	Pilot	High	28	20/20	Not relevant

More detailed information about each requirement specification case is included in Annex 1.

For each of these requirement specification cases, evaluation of the experiences was based on the aspects and criteria defined in section 4.2 Piloting feedback collection method.

4.2 Piloting feedback collection method

Experiences about the new proposed methods were collected from the pilot requirement specification cases by interviewing the requirement specification author, Use Case workshop facilitator and test responsible. In sim-

ilar manner, experiences from the reference cases were collected by interviewing the reference requirement specification author and test responsible.

Feedback from reference and pilot requirement specification cases was collected for different categories related to each of the enhancement proposal. The categories and evaluation criteria are shown in Table 6 Evaluation categories and criteria.

Table 6 Evaluation categories and criteria

Evaluation category	**Criteria**	**Relevant cases**
1. Feature team experiences:	**Feature team related aspects that have been considered in this study are the following:**	
1.1 Expertise coverage	It is highly important for a well functioning feature team that experts from each involved expertise areas can be involved. Thus expertise coverage successfulness for each of the pilot requirement specification cases is evaluated.	Cases 3 – 4
1.2 Participant activity	Feature team participants provide valuable input form their own expertise areas in the Use Case scoping, as well as support the potential problem solving in later phases. For the pilot requirement specification cases, the main emphasis is in evaluating the participant activity for the Use Case scoping e.g. during Use Case workshop.	Cases 3 – 4
2. Use Case scoping experiences:	**Use Case scoping related aspects that have been considered in this study are the following:**	
2.1 Use Case scope and coverage	Use Case scope and coverage successfulness is evaluated for each requirement specification: how well the chosen Use Case set fits to customer demand (content and schedule), how well feature interworking can be covered, and how well the chosen Use Case set is expected to support test planning.	Cases 1 – 4
2.2 Use Case quality	Use Case quality is evaluated with the following criteria: Use Case descriptions are clear, precise and correct. Use Case descriptions are also directly usable for test case descriptions.	Cases 1 – 4
3. Use Case workshop experiences:	**Use Case workshop related aspects that have been considered in this study are the following:**	
3.1 Participant availability	Participant availability evaluation is based on how well all the involved expertise areas are represented in the Use Case workshop.	Cases 3 – 4
3.2 Workshop efficiency	Use Case workshop can be evaluated to be efficient, when a set of Use Cases that will be developed further can be chosen without need to arrange further workshops.	Cases 3 – 4
3.3 Competence transfer success	The most important non-tangible result of the Use Case workshop is the increased competence level of the all involved persons. Competence transfer successfulness evaluation is done based on Use Case work-	Cases 3 – 4

	shop facilitator experiences.	
4. Testing quality related experiences:	**Testing quality related aspects that have been considered in this study are the following:**	
4.1 Test coverage	Test coverage evaluation is based on how many test cases have been created based on the Use Cases and requirements of the new functionality, and on estimation how well the overall functionality can be covered with the defined test case set.	Cases 1 – 4
4.2 Testing viewpoint	For testing viewpoint, evaluation is done based on estimation how well the actual customer requirements can be covered in the testing instead of testing what is the implementation (customer scope vs. R&D scope).	Cases 1 – 4

In addition to the detailed comments and feedback, a grade was requested from the requirement specification author, from Use Case workshop facilitator and from test responsible for each of the categories. The given grades are used for quantitative comparison between the reference and pilot requirement specifications.

For the reference requirement specifications, the feature team and Use Case workshop experiences are not relevant. In both of the piloted requirement specifications all the three enhancement proposals were present and available for the evaluation, i.e. Use Case scoping process, Use Case workshops and virtual feature teams. Virtual feature team concept is not yet in general use in the NSN mobile voice solution area, and because of this, for both piloted requirement specification a corresponding virtual feature team was temporarily formed with participants from all related parts of the organization.

The detailed feedback comments for each requirement specification case are included in Appendix 1 Detailed feedback from the reference and pilot cases. The results are summarized in section 4.3 Piloting result analysis.

4.3 Piloting result analysis

The given grades for each requirement specification case and evaluation category are collected for summary and comparison in the table 7 Grade summary.

A scale from grade 1 to grade 5 is used. Grade 5 represents the highest value, when all the targets for the category can be considered to be successfully achieved according to the defined criteria in the table 5 Evaluation categories and criteria.

Table 7 Grade summary

	Grade			
	Case1	**Case 2**	**Case 3**	**Case 4**
1. Feature team experiences:				
1.1 Expertise coverage	Not relevant	Not relevant	5	3
1.2 Participant activity	Not relevant	Not relevant	4	3
2. Use Case scoping experiences:				
2.1 Use Case scope and coverage	3	3	5	4
2.2 Use Case quality	2	3	4	3
3. Use Case workshop experiences:				
3.1 Participant availability	Not relevant	Not relevant	4	4
3.2 Workshop efficiency	Not relevant	Not relevant	4	3
3.3 Competence transfer success	Not relevant	Not relevant	5	4
4. Testing quality related experiences:				
4.1 Test coverage	Not available	3	Not available	Not available
4.2 Testing viewpoint	Not available	2	Not available	Not available

"Not available" is indicated in the test coverage and testing viewpoint categories, if information is missing because the test planning wasn't yet started. Correspondingly, "Not relevant" is indicated when the category is not relevant for the requirement specification case.

Based on the grades, two main comparisons can be done:

1. Use Case scoping experiences in the pilot requirement specification cases compared with the experiences from the reference pilot specifications.
2. New method usage experiences compared between the two pilot requirement specifications. The new methods include feature team and Use Case workshop usage experiences.

In addition to these two grade based comparisons, several other generic remarks can be drawn based on the detailed comments from the pilot and reference cases.

Use Case scoping experiences

The given grades for the Use Case scope and coverage (Category 2.1) and Use Case quality (Category 2.2) for all included reference and pilot requirement specifications are shown in the figure 11 Use Case scoping experience comparison below.

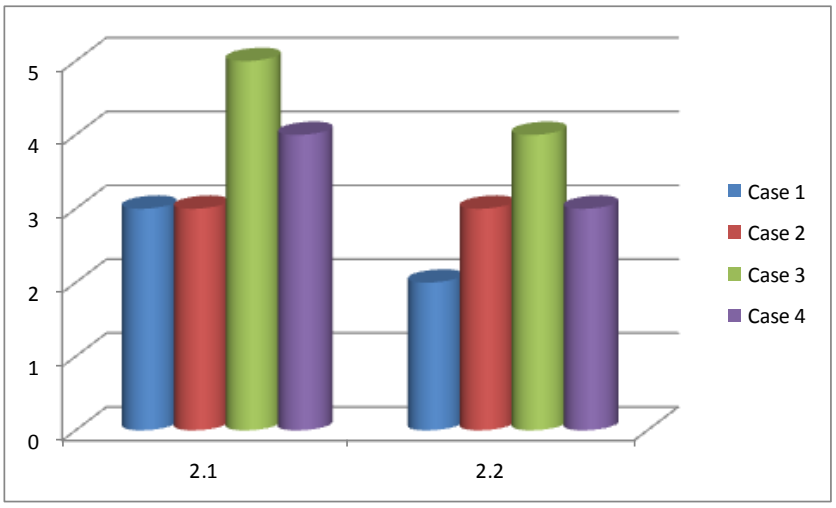

Figure 11 Use Case scoping experience comparison

From the graphics it can be seen that the Use Case scoping and quality has been evaluated to be overall better in the pilot cases (Case 3 and Case 4), than in the reference cases (Case 1 and Case 2). Especially when comparing Case 1 and Case 3, which are both medium complexity level requirement specifications, the difference is significant in favor of Case 3. However, when the high complexity Case 2 and Case 4 are compared, it can be seen that the Use Case scoping has been more successful in Case 4, but the Use Case quality is at the same level in both of the cases.

The difference between Case 3 and Case 4 Use Case quality results may result from several factors. One of these is naturally the difference in the complexity level. When the overall environment and feature scope is very complex and maybe even partially undefined at the starting phase, it is for example more difficult to have commitment from all the related areas into the feature teams. Due to insufficient commitment, the full benefits of the new methods may not be reached.

Based on the results it can be seen that the new methods are overall beneficial for the Use Case scoping and quality. However, as prerequisite before entering the requirement specification phase, it needs to be ensured that the business requirements are clear enough. Sufficient commitment level has to exist from all related areas, which are needed for the feature team.

New method usage experiences

Feature team experiences (Categories 1.1 and 1.2) and Use Case workshop experiences (Categories 3.1, 3.2 and 3.3) for the two pilot requirement specifications are shown in the figure 12 New method experience comparison below.

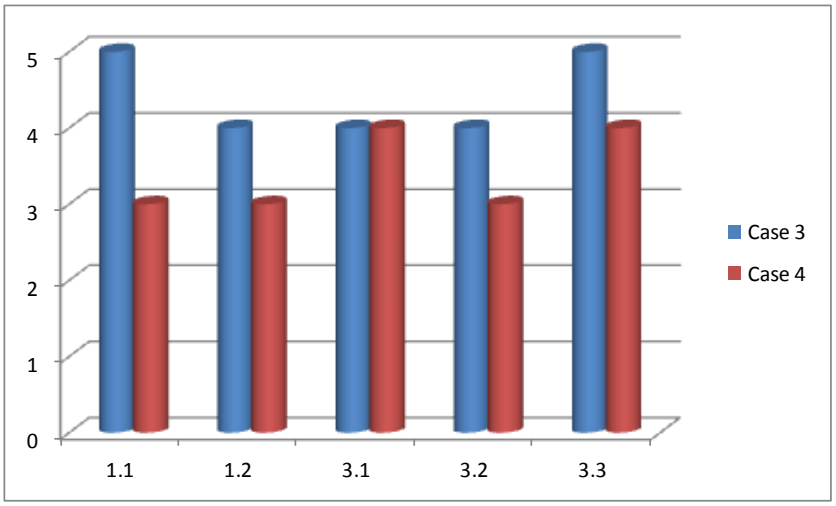

Figure 12 New method experience comparison

When comparing the feature team related experiences (Categories 1.1 and 1.2) between the two pilot requirement specification cases, it is seen that in the medium complexity Case 3 the experiences have been better than in the high complexity Case 4. The main explanation for the difference is, that in Case 4 it was more challenging to form the feature team with committed team members from several related product lines than in Case 3 with lower complexity level.

The Use Case workshop experiences (Categories 3.1, 3.2 and 3.3) are generally encouraging. The Use Case workshop participation (Category 3.1) in both pilot cases has been overall quite good. However, the Use Case workshop efficiency (Category 3.2) has been better in Case 3. The main reason again seems to be the more unclear scope and complex environment in Case 4, which has reduced the participant activity during the Use Case workshop. Despite of this, the competence transfer success (Category 3.3) in both Case 4 and Case 3 has been very good.

Generic remarks

Overall, requirement definition via the established feature teams and Use Case workshops seems to contribute to the given targets in the expected manner. Use Case screening has helped to concentrate to a smaller set of Use Cases in both of the pilot requirement specification cases. As result, this reduces the amount of such Use cases and requirements that would have been set into passive state, and this way "nice to have" requirements are reduced.

The successfulness of the new methods is dependent on e.g. how well the problem area is understood in the first place, are the business requirements clear, what is the complexity level of the issue, and are the commitments from the related areas and experts available.

In the high complexity requirement specifications the Use Cases tend to be defined at higher detail level than in the medium complexity requirement specifications. This is quite natural, and after the solution area becomes clearer, additional studies can be carried out.

The Use Case workshop efficiency depends on the quality of workshop inputs, as well as on participant availability and activity. If the preparation for the Use Case workshop is insufficient or if feature team key contributors are missing from the workshop, then it is not possible to make decision on the chosen Use Case set. This results into need for additional meetings, which reduces efficiency.

4.4 Analysis of the used method

In the piloting phase, it was possible to include only two requirement specification cases within the given timeframe. Therefore, even the experiences are good and support the new practices for requirement definition, it would have been beneficial to have experiences from a larger amount of pilot cases. The main obstacle of including additional pilot requirement specifications was that the schedules of ongoing system requirement specifications were already demanding. Utilizing new methods would have taken more time and thus it could have pushed the requirement specification completion further, which was not seen reasonable.

The evaluation of each pilot and reference case was done via the set criteria for each evaluation category, as presented in table 4. Even the given grades were based on the defined criteria; they are still subjective measures evaluated by the involved requirement specification authors, Use Case workshop facilitators and testing responsible persons. Grading of the experiences in each category was, however, seen as the most efficient way for comparing the results from the requirement specification cases. The reason is that there were no such official and suitable tools with quantitative or qualitative measurements in place, which would have enabled more objective result analysis and comparison.

It was also a challenge to find suitable measures and criteria for comparing the already existing reference requirement specification cases with the piloted requirement specification cases. For this purpose, the Use Case scoping successfulness, and the experienced Use Case quality was expected to provide the best match for comparison. Naturally, the successfulness of the new requirement definition practices for Use Case workshops and feature teams could be compared only between the pilot requirement specification cases, which were using these practices.

In this study, mainly the requirement specification phase successfulness with the new methods has been analyzed. Based on these results it is not yet possible to draw conclusions e.g. about a potentially decreased amount of faults found later in the product creation process or about improved testability. However, because more efficient Use Case screening practice is introduced, only the most important Use Cases are implemented in the future for certain release or iteration, and thus there should not be as many

source possibilities for faults as when all the Use Cases would have been worked forward without careful selection.

During the pilot and reference requirement specification comparison, it became evident that the transition to end-to-end Agile and Iterative product development is just in the beginning at NSN voice solution area. The requirement specifications are still in practice prepared prior the implementation phase is entered, or well prior to when the test planning is started. For example, analysis of the pilot requirement specification case's testability, e.g. based on the amount of test cases was impossible, because test planning was not yet done in the timeframe of this study for most of the cases.

4.5 Recommendation

Based on the results and collected experiences, it is recommended to implement the new practices of Use Case scoping, Use Case workshops and feature teams as part of the Agile and Iterative requirement definition in NSN's mobile voice solution area.

In very high complexity requirement specification cases it may be necessary to consider case by case, if the requirement specification already benefits about the new practices. It may be more efficient to carry out a preliminary study first to clarify the scope and business requirements, before the actual requirement specification phase with the new practices is entered into.

5 GUIDELINES FOR AGILE AND ITERATIVE REQUIREMENT DEFINITION

In this section as part of the practical work of this study, further guidelines are given for Agile and Iterative requirement definition. The guidelines cover the three new introduced enhancements for requirement definition process: feature team establishment and guidelines for Use Case workshop practices and for Use Case screening. Note that additional role level instructions need to be created later, if it is decided that the enhancements are taken into use.

5.1 Feature team establishment

To be able to utilize the new methods, a long term feature team establishment is not mandatory; corresponding expertise can also be collected from the different parts on the organization on need basis.

In practice, it is necessary to evaluate for each new requirement specification the required participants and contacts for the work. Participants of such virtual feature team should consist of e.g. the following representatives:

- Feature responsible
- Requirement specification responsible
- Business responsible
- Requestor of the feature and/or customer representative
- Software design and implementation responsible(s)
- Verification responsible(s)
- Project management responsible(s)
- Counterpart product representatives (when relevant)
- Platform implementation responsible (when relevant)
- Interworking feature responsible (when relevant)
- Network planning representative (when relevant)

Feature team participants need to be involved in all information sharing related to the new feature or functionality, as well as be invited to the Use Case workshops.

5.2 Use Case workshop guidelines

The purpose of the Use Case workshop is to preserve a timeslot for the virtual feature team participants to define, discuss and decide the required Use Cases of the new feature or functionality. The aim is to conclude on a screened set of high-quality Use Cases.

5.2.1 Use Case workshop preparation

Even the Use Case workshop is arranged to define the new feature or functionality Use Cases in a collaborative manner, certain preparations should be done to enable a successful workshop.

First of all, the scope of the new feature or functionality on business level should be clear enough to enable drafting of the Use Cases. If this is not the case, the scope should be clarified e.g. by lead of the feature responsible, business responsible person, requirement specification responsible, or even in co-operation by all of them. The customer requirements need to be made clear, and optionally acceptance for the planned content could be asked directly from the customer or in some cases from customer teams.

For example, User Stories could be helpful in clarifying the business requirements. User Stories describe why the new feature or functionality is needed, who will use it, and what outputs are expected from the system. Further clarification could be sought via high level examples to better see how the feature will be useful for the customer. To find out which aspects of the new functionality are the most important ones, an alternative solution or workaround solution could be asked, if customer would not have this feature. This ensures that a second thought to the issue is paid and the most obvious solution is not accepted automatically.

After the scope of the functionality is clear, the requirement specification writer can start drafting the Use Case descriptions on high level. Having the draft Use Case descriptions available will fasten the process within the Use Case workshop. However it is not feasible to add too many details into the draft Use Cases, so that the Use Case workshop participants don't just start to check the already defined content, but are enabled to discuss freely the purpose and functionality of each Use Case.

While the actual Use Case workshop reservations are being done, a recommended length for a workshop duration is typically a couple of hours. If the new feature is very complex, it may be beneficial to divide the Use Case workshop into for example two different occasions due to participant availability. In most cases it may be very difficult to get experts from different areas to participate into a workshop that lasts more than maximum of three hours. It is good from the participant activity viewpoint too, that the actual workshop event is not too long.

Use Case workshop should have a facilitator, who could be e.g. the feature responsible, business responsible or verification responsible, however it would be beneficial if the facilitator is not the requirement specification responsible. This way the requirement specification responsible is fully enabled to follow and capture the comments and proposals made in the workshop, without a need to steer the workshop forward which is the task of the workshop facilitator.

The invited participants should include the representatives of the feature team; however for optimal work efficiency maximum participant amount should not exceed 10 – 12 persons. Depending on the new functionality or feature, it may be beneficial to divide the participants into smaller groups within the workshop to discuss certain area Use Cases.

In the figure 13 below, the main tasks in the Use Case workshop preparation phase are shown. On the left hand side of the figure, the main contributor for each step is indicated, and on the right hand side the main purpose of the step is summarized.

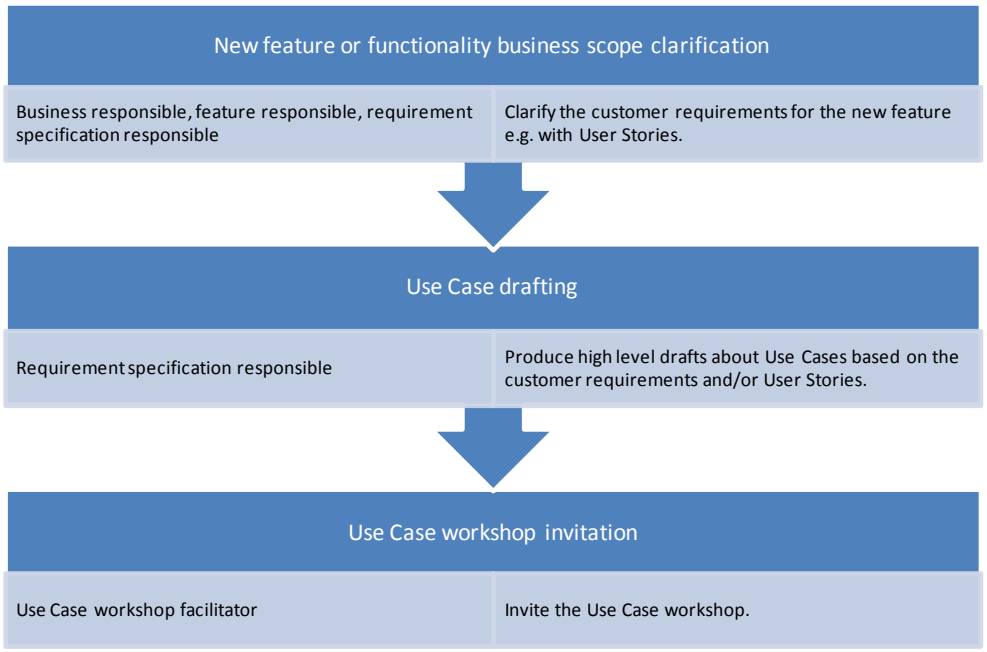

Figure 13 Use Case workshop preparation steps

5.2.2 During Use Case workshop

In the beginning of the Use Case workshop it is a good practice to clarify for the participants, what the Use Case workshop is all about, and what it is not. In the Use Case workshop the Use Cases are discussed and defined collaboratively and actively between all the participants of the feature team. Discussion should be on level that "what" software does, not on "how" it is implemented. There should not be detail discussions about how the new functionality should be implemented, and if such discussion session is needed, it should be held separately. Identifying the scope of the Use Case workshop is important especially in the beginning when the new practice is taken into use and people are not yet familiar with the concept.

After clarifying the Use Case workshop purpose, introduction to the new feature or functionality is needed. Feature responsible, business responsible or requirement specification responsible should describe what the system is expected to do and how the feature should work when it has been implemented. Practical examples should be used. For this purpose, the User Stories are a good tool to illustrate the related end-to-end functionality and what is trying to be achieved with the new feature or functionality. At this phase, it is important that other participants ask questions to make the overall functionality clear and to find out all additional interworking and

edge cases. Especially implementation and verification experts have the main role in this clarification phase.

At the point when the overall required functionality is clear for all participants, it is possible to start discussion about the actual Use Cases. As indicated in the Use Case preparation phase guidelines, it is beneficial that e.g. requirement specification responsible prepares a set of draft Use Cases as basis for the discussion. In practice, during the workshop notes are collected by the requirement specification responsible for the draft Use Cases, which will be fine-tuned according to the discussion and comments given by the workshop participants. Actual Use Case modification and editing should happen off-line after the Use Case workshop event. It is possible that some of the drafted Use Cases are dropped and new, more appropriate ones are identified and included instead during the workshop. In order to get everyone actively involved and to encourage open discussion in this phase, specific questions about the Use Cases could be asked from different angles by e.g. the workshop facilitator or the feature responsible.

Once the required Use Cases have been agreed, it should be discussed what kinds of Use Case groups are formed to be able to prioritize the Use Cases. This is also called as Use Case screening, for which further guidelines are given in section 5.3 Use Case screening guidelines.

At the end of use case workshop, all participants should feel comfortable to move on based on the identified Use Cases and Use Case groups. If some scenario or Use Case is not clear enough, additional information has to be sought from e.g. business responsible, customer team or from the customers directly. The facilitator needs to ensure that all the potentially discovered open issues have been captured and that there is a responsible person to clarify the issues. Everyone involved deserves a big “Thank You” about the active contribution and of the job well done.

The following figure 14 shows as an overview the main steps of a Use Case workshop. On the left hand side of the figure, the main contributor for each step is indicated, and on the right hand side the purpose of the step is summarized.

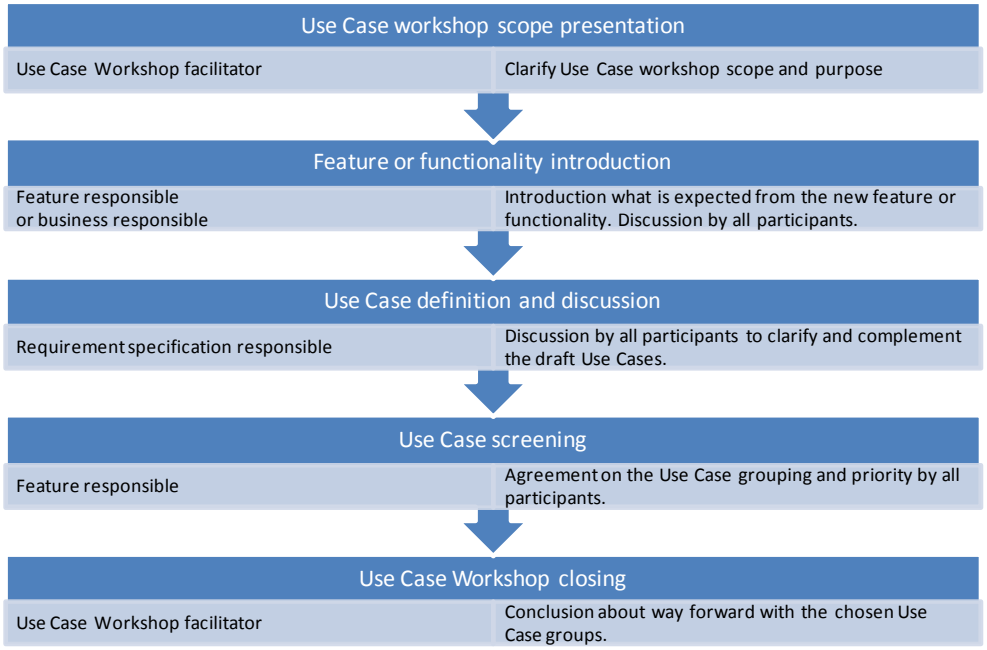

Figure 14 High level phases of Use Case workshop

5.2.3 Outcome from the Use Case workshop

The outcome deliverables from the Use Case workshop are the prioritized list of Use Cases and the draft Use Case descriptions themselves.

An example of the Use Case list format is shown in the figure 15.

UC number	UC name	Subgroup	Business priority	Technical difficulty	Verification impact	Complexity for customer	UC group	Comments
UC001	xxxx	xxxx	1	Low	Medium	High	UG001	xxxxx
UC002	xxxx	xxxx	2	Medium	High	Low	UG002	xxxxx
UC003	xxxx	xxxx	3	High	Low	Medium	UG003	xxxxx

Figure 15 Example of a Use Case list

The Use Case list documents and summarizes the common understanding of all involved parties about the required Use Cases and their priorities for the new feature or functionality. As discussed earlier, only the most important Use Cases are further defined, implemented and verified in the first release or iteration. The lower priority Use Cases will be evaluated correspondingly for implementation in forthcoming releases or iterations.

The Use Case list should be stored together with the other new feature or functionality related materials, where everyone in the feature team has access. The Use Case list could be linked to the requirement specification document as well.

Other outcome deliverables from the Use Case workshop are the draft Use Cases with brief descriptions. In practice, these are typically included in the draft requirement specification document, according to the existing requirement specification template. Draft Use Case descriptions are "raw material" for the following requirement specification work, during which the Use Cases are further defined in more details and the actual requirements are derived from the use cases by the requirement specification responsible.

As a non-tangible outcome from the Use Case workshop a common understanding of the problem and new functionality by all parties is expected to be achieved. This will help in all future communication related to the new functionality and it should lead to finding potential faults earlier. Any communication and presentation material distributed during the Use Case workshop, in order to increase the competence of the feature team, should be stored together with the other new feature or functionality related materials where everyone in the feature team has access.

5.3 Use Case screening guidelines

When the draft Use Cases for the new feature or functionality have been agreed, Use Cases are prioritized and grouped as discussed earlier, to enable the decision making on what is the suitable set of Use Cases for certain releases and iterations.

The following aspects for each Use Case should be considered during the Use Case workshop, for example:

- Business priority (e.g. on scale of 1 – 3, where 1 is highest priority), proposed by the business responsible.
- Technical difficulty (e.g. on scale low/medium/high), proposed by the implementation experts.
- Verification impact (e.g. on scale low/medium/high), proposed by the verification experts.
- Complexity for customer (e.g. on scale low/medium/high), proposed by e.g. the feature responsible or customer (team) representative.

Based on the proposals, further discussion may take place to reach a common agreement on each aspect. At this point, the Use Cases themselves are already known in details by all participants of the Use Case workshop, and therefore it is actually quite fast and intuitive process to go through the Use Cases for the different above mentioned aspects. For example, spending few minutes with each Use Case is likely to be sufficient.

Once the above aspects for each Use Case have been agreed, the Use Case grouping can be done. The main input is the business priority. As main rule, Use Cases of the same priority should be included in one Use Case group. In practice, manual tuning of the Use Case group is needed. For example, if some of the Use Cases are extremely complex to implement or the verification in the given timeframe is not possible, then those Use Cas-

es need to be postponed to later Use Case group. Some lower business priority Use Cases may be beneficial to be included in a higher priority Use Case group, in case there are implementation synergies with the other higher priority Use Cases. Overall, the Use Case groups need to form entities, which make sense from business, implementation and verification viewpoints. They all need to provide clear added value for the customer in the given timeframe.

Use Case screening is actually the most fun and rewarding phase typically; it is seen how the teams' jointly defined and chosen Use Case groups match and contribute to the real customer demand.

An example of Use Case list is given in the section 5.2.3 Outcome from the Use Case workshop, figure 15 Example of a Use Case list.

6 SUMMARY AND CONCLUSIONS

A strong demand exists at NSN voice solution area to change the current product creation model towards end-to-end agile development. Currently agile development model is used during implementation and functional testing phases of product development. In contrast, the requirement definition and the system verification phases are in practice still using the traditional Waterfall model. As described earlier, requirement definition is in the key role in the success of the new feature or functionality. Therefore, enhancements were studied to enable the Agile and Iterative requirement definition. This in turn is expected to help to achieve full end-to-end agile development model, and to fulfill the successful incremental software delivery need.

In this study, the main drivers at NSN voice solution area for changing the requirement definition methods to better support the end-to-end Agile and Iterative product creation were studied. As result of the theoretical study, three enhancement proposals were identified: introduction of feature teams, arranging Use Case workshops and carrying out more detailed Use Case screening for the new feature or functionality. In the practical part of the study, a piloting phase was arranged with real requirement specifications to test the enhancement proposals in practice. Based on the theoretical analysis and the gained results from the pilot cases, a recommendation to implement the enhancements at NSN voice solution area was given. Finally, guidelines are provided on how these new enhancements can be used in practice.

Overall, the targets that were set for the study were achieved. As a requested output, the enhancement proposals for requirement definition were provided. Due to the timeframe of the study, the amount of included pilot requirement specifications was quite modest and therefore the amount of practical experiences was quite small. However, the results from the completed pilots were encouraging, and based on them it was possible to recommend to take the enhancement proposals in use for requirement definition. In addition, guidelines for the Agile and Iterative requirement definition were given.

6.1 Further development items

During this study it was possible to analyze and reveal some of the pain points and obstacles in requirement definition phase that need to be removed when true end-to-end Agile and Iterative product creation model is entered. There however is a long journey still ahead and therefore some further improvement possibilities are given below.

It needs to be noted that there are several other development activities ongoing in parallel at NSN voice solution area due to the ongoing change towards end-to-end Agile and Iterative product creation model. On the other hand, to reduce the complexity and to divide the work into manageable pieces, this study intended to only cover the improvements for the requirement definition phase. It is possible, that the enhancement proposals

presented in this study, may be further adjusted according to the needs of the overall change towards the end-to-end Agile and Iterative product creation model. For example, Automated Test Driven Development may be later entered into. From this background, the proposed requirement definition improvements have been chosen so, that they are compatible with other potential future improvements too.

A potential future development topic is to include better measurement possibilities in relevant requirement and test case management tools. This would allow tracing of a customer requirement all the way up to a verified acceptance test case, in order to help in revealing efficiency and quality issues.

In full end-to-end Agile and Iterative development, the different product creation phases should be able to proceed in parallel. The parallel working model is enabled with the proposed enhancement to requirement definition phase, but mindset, resource allocation and product management change is needed to fully transit into end-to-end agile model. The mindset change in smaller scale could be tried in beginning. Parallel working could be trialed, for example, when a proof-of-concept about a new solution for a customer would need to be delivered within a very limited timeframe.

PREFERENCES

Abrahamsson P., Salo O., Ronkainen J., Warsta J. 2002. Agile software development methods: Review and analysis. VTT

Adzig, G. 2009. Bridging the Communication Gap. Neuri Limited

Adzig, G. 2011. Specification by example. Manning Publications Co.

Agile Alliance. 2012. Accessed 26.6.2012. http://www.agilealliance.org/

Awad, M.A. 2005. A Comparison between Agile and Traditional Software Development Methodologies. Accessed 30.8.2012. http://pds10.egloos.com/pds/200808/13/85/A_comparision_between_Agile_and_Traditional_SW_development_methodologies.pdf

Cockburn, A. 1999. Writing Effective Use Cases. Accessed 19.5.2012. http://www2.dis.ulpgc.es/~jsanchez/MDS/EffectiveUseCases.pdf

Cockburn, A. 2000. Agile Software Development. Addison-Wesley

Craddock A., Richards K., Tudor D., Roberts B., Godwin J. 2012. The DSDM Agile Project Framework for Scrum. DSDM Consortium. http://www.dsdm.org/wp-content/uploads/2012/05/The-DSDM-Agile-Project-Framework-v1-1.pdf

Get It Right the First Time: Writing Better Requirements. 2008. Telelogic DOORS. IBM Corporation. Accessed 30.8.2012. http://publib.boulder.ibm.com/infocenter/rsdp/v1r0m0/topic/com.ibm.help.download.doors.doc/pdf/get_it_right_the_first_time.pdf

Gottesdiener, E. 2006. Requirements by Collaboration. Pearson Education Inc.

Hoegl, M., Gemuenden, H. 2001. Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept and Empirical Evidence. Organization Science, Vol. 12, No. 4. (Jul. - Aug., 2001)

Kalermo, J., Rissanen, J. 2002. Agile Software Development in Theory and Practise. Master's Thesis. University of Jyväskylä

Larman, G. 2007. Agile & Iterative Development. Pearson Education Inc.

Larman, G., Vodde, B. 2010. Feature Team Primer. Accessed 19.5.2012. http://www.featureteams.org/

Saft, F. 2012. Improving the Agile Methods and Open Source Lab Course. Master Thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg

Waterfall Model. 2012. Accessed 22.5.2012. http://www.waterfall-model.com/

Appendix 1/1

DETAILED FEEDBACK FROM THE REFERENCE AND PILOT CASES

Case 1: Multi-Operator Core Network (MOCN) support in MSS (Reference)

Case 1 is a system level requirement specification which was done without the proposed enhancements in the requirement specification phase. Therefore the requirement specification author has been in the main role for defining the Use Case set and corresponding system and product level requirements. Consultancy from related implementation experts was requested on need basis, however wider group of participants was included only for the requirement specification review.

The new feature itself is a 3GPP standard network sharing solution and therefore the scope was quite clear.

This requirement specification contains:
- System level Use Cases: 11 active and 4 passive
- System and product requirements: 14 active and 1 passive
- Impacts to external interfaces: 2

Test cases planned: Not available (Test case planning not yet started)

Table 8 Experiences from Case 1

Evaluation category	Experiences	Grade
1. Feature team experiences:		
1.1 Expertise coverage	Not relevant	Not relevant
1.2 Participant activity	Not relevant	Not relevant
2. Use Case scoping experiences:		
2.1 Use Case scope and coverage	15 Use Cases are included in total, 4 of them are intended for future enhancements and are marked as passive. The chosen Use Case set is expected to cover the 3GPP standard requirements and meet the required customer schedule well. No customer specific Use Cases are identified and included. In the Use Case set interworking scenarios with certain related features are covered, however it can't be evaluated which interworking features may have been potentially missed. The Use Cases have been divided into small functionality steps, however no optimal verification support can be expected because for complete end-to-end functionality several Use Cases may need to be combined. Use Cases seem to cover the requirements, but it is not visible that what operator can and would like to do with the feature overall. I.e. path from business requirements to implementation requirements is lost.	3
2.2 Use Case quality	The Use Cases have been described on detailed level. However, in the Use Case descriptions also implementation suggestions are given, which makes the descriptions less clear than they could be. In practice, implementation specification has to be read to find out what is finally required to be verified. Because the Use Cases have been split into small functionality steps, test planning based on these Use Case descriptions can't be done as one to one mapping.	2
3. Use Case workshop experiences:		
3.1 Participant availability	Not relevant	Not relevant
3.2 Workshop efficiency	Not relevant	Not relevant
3.3 Competence transfer success	Not relevant	Not relevant
4. Testing quality related experiences:		
4.1 Test coverage	Test case planning has not yet been started.	Not available
4.2 Testing viewpoint	Test case planning has not yet been started. In end-to-end verification the scope should be in the customer requirement validation, whereas in the other testing phases e.g. functional testing the implementation correctness is checked.	Not available

Case 2: IPv4/IPv6 dual stack in MSS system (Reference)

Case 2 is a system level requirement specification done without the proposed enhancements in the requirement specification phase. As in Case 1, the requirement specification author has been in the main role.

Technically the requirement specification is related to basic connectivity improvements. It contains Use Cases and requirements for IPv6 support for SIP sessions and IPv4/IPv6 interworking at several interfaces of MSC Server system. The purpose of the feature is to introduce and verify commercial level IPv4/IPv6 interworking capability in MSC Server system.

Additional challenge is that the whole IP connectivity and network infrastructure has to be IPv6 capable on several signaling layers.

This requirement specification contains:
- System level Use Cases: 12 active and 0 passive
- System and product requirements: 34 active and 0 passive
- Impacts to external interfaces: 8

Test cases planned: 28

Table 9 Experiences from Case 2

Evaluation category	Experiences	Grade
1. Feature team experiences:		
1.1 Expertise coverage	Not relevant	Not relevant
1.2 Participant activity	Not relevant	Not relevant
2. Use Case scoping experiences:		
2.1 Use Case scope and coverage	From functionality viewpoint it is expected that the Use Case set covers well the customer expectations. However the network management aspects may not fully covered because the operator expectations at this area are not known in detail. This is an extremely large area and building the own specific test environment is a huge task with new equipment. There have been some delays in the schedule of IPv4/IPv6 interworking introduction in MSC Server system, but the current schedule is still quite well aligned with e.g. IPv6 capable terminal availability. Further development work is needed in later phases (e.g. for maintenance and operability as well as for internal MSC Server system interfaces for IPv6 support). In this phase the goal has been in the planned functionality included already in the requirement specification. Additional feature interworking needs and issues may be found later, depending on the support of related MSC Server system external network elements. There is no clear visibility to other organizational areas which implement the corresponding capability to the related network elements. Use Cases have been written from testing viewpoint, and thus it is expected that use cases support the verification well. Most use cases are end-to-end level cases and thus quite large.	3
2.2 Use Case quality	Large end-to-end use cases are handled quite shortly in the requirement specification and therefore all possible details are likely not covered. It is expected that further clarifications before verification is started are needed.	3
3. Use Case workshop experiences:		
3.1 Participant availability	Not relevant	Not relevant
3.2 Workshop efficiency	Not relevant	Not relevant
3.3 Competence transfer success	Not relevant	Not relevant

Table 9 Experiences from Case 2 (cont.)

4. Testing quality related experiences:		
4.1 Test coverage	Successful test case coverage based on the use cases is expected to be good, however failure cases and configuration error related cases are not included sufficiently in the test case set.	3
4.2 Testing view-point	This feature concentrates on the generic functionality improvements of MSC Server system, and into testing of its' already existing functionality. Due to the nature of the feature (basic connectivity capability related) no clear input from customers has existed. Thus customer specific requirements are not known and the test scope also concentrates to the internally decided content.	2

Case 3: MSS Pool – enhanced control for LTE/CSFB and LA off-loading (Pilot)

Case 3 is the first MSC Server system level requirement specification which was done according to the enhancement proposals. This requirement specification includes a screened set of system level Use Cases, and corresponding system and MSC Server product requirements.

MSS pooling solution is one of the most customer attracting features at the moment at network resiliency area. Based on received customer feedback, the new feature adds functionality to better manage the subscriber distribution among the pooled MSSs in 2G/3G and LTE networks.

This requirement specification contains:
- System level Use Cases: 11 active and 14 passive
- System and product requirements: 20 active and 0 passive
- Impacts to external interfaces: 4

Test cases planned: Not Available (Test case planning not yet started)

Table 10 Experiences from Case 3

Evaluation category	Experiences	Grade
1. Feature team experiences:		
1.1 Expertise coverage	Participants from all the relevant teams were available. Testing organization participation even exceeded the expectations.	5
1.2 Participant activity	The overall participant activity was good. Main challenge was to get all the people present in the workshops at the same time. Especially product management / business representatives are quite busy overall.	4
2. Use Case scoping experiences:		
2.1 Use Case scope and coverage	25 Use Cases are included in total, 14 of them have been screened out from the scope and are marked as passive. Use case coverage is better than would have been possible with earlier methods. By estimation ~50% more use cases were found as result of team effort, compared to situation if requirement specification writer would have collected them alone. For example, better understanding of customer feedback resulted into additional Use Cases. Feature interworking is better ensured: in the multi-expertise team, many interworking features came up which otherwise could have been missed. From the total Use Case set, only the highest business value Use Cases were chosen to be worked forward. Thus the requirement specification scope is much better aligned also with business need and implementation / verification possibilities and less waste can be expected because lower priority use cases are not worked forward in this phase yet.	5
2.2 Use Case quality	Use Case quality is improved: due to involved verification expertise and active contribution, Use Cases can be more easily used as basis for test planning. Use Case vs. Test Case mapping is almost one to one. Use Cases are written without confusing information about e.g. implementation alternatives.	4

Table 10 Experiences from Case 3 (cont.)

3. Use Case workshop experiences:		
3.1 Participant availability	Participants from the all relevant teams were available for the Use Case workshops, however product management representatives were not able to participate the full time due to other meetings. Even the Use Case workshop was split into two different occasions, the overall participation was good.	4
3.2 Workshop efficiency	Use Case workshop was split into two different occasions. The technical aspects for the Use Cases were possible to be discussed and agreed during the first arranged Use Case workshop (3 hours), however a second (1 hour) Use Case workshop had to be arranged for screening the Use Case set.	4
3.3 Competence transfer success	As result of the Use Case workshops, everyone involved were familiar with the new feature content in the early requirement specification phase and are able to start working on it, or at least make planning and preparation for the later product creation phases. Positive feedback has been received in practice from all the participants. Better understanding about the feature also helped to understand test environment needs and effort estimations of the later product creation phases.	5
4. Testing quality related experiences:		
4.1 Test coverage	Test case planning is not started yet.	Not available
4.2 Testing viewpoint	Test case planning is not started yet. At this phase it is however felt that it will be easier to focus to customer requirement verification than based on Use Cases defined with earlier methods.	Not available

Case 4: Georedundancy solution for VoLTE (Pilot)

Case 4 is a pilot MSC Server system level requirement specification which was done by utilizing the enhancement proposals during the Use Case definition phase. The requirement specification includes a screened set of system level Use Cases, but does not introduce new system or product requirements. This is because the purpose of this requirement specification was to collect together the current VoLTE related geo-redundancy related capabilities at system level to enable end-to-end verification.

Geo-redundancy is an essential functionality in the network and is considered as a basis for commercial VoLTE deployments. Providing a geographical redundancy for the users of an IMS based VoLTE solution, several system level approaches to achieve this geographical availability have been studied.

Therefore this requirement specification covers a very large system level solution involving several products owned by different product lines for the full NSN VoLTE solution, which makes the requirement specification work more complex compared to normal MSC Server system related requirement specifications.

This requirement specification contains:
- System level Use Cases: 15 active and 0 passive
- System and product requirements: Not relevant
- Impacts to external interfaces: 28

Test cases planned: Not Available (Test case planning not yet started)

Table 11 Experiences from Case 4

Evaluation category	Experiences	Grade
1. Feature team experiences:		
1.1 Expertise coverage	Testing expertise representation has been good. Product management has been involved, however more active participation in the beginning of the process e.g. for feature scoping would have been needed. Very good support has been received from some of the co-product areas, but from some co-product areas support has been minimal. Commitment to requirement specification support from all areas was not sufficient.	3
1.2 Participant activity	Input to Use Cases was received from several sources, but in a quite uncontrollable manner and mainly outside of the Use Case workshop. Main support for Use Case creation and prioritization was received from the product management, but the main effort of the Use Case set collection and creation was still on the requirement specification author. Use Case prioritization was done during the testing strategy planning, because verification expert availability was possible only at the end of the process. It would have been good to have better commitment from the different areas from the start, to enable more active participation. Preparation in general was not sufficient by the participants, even though instructed before the Use Case workshop.	3
2. Use Case scoping experiences:		
2.1 Use Case scope and coverage	VoLTE geo-redundancy is the main scope of this feature. The defined Use Cases cover the scope well. Interworking to other related features was covered. From testability viewpoint, with the Use Case prioritization the scope is ok. Work share between end-to-end verification and functional testing was clarified during the Use Case prioritization.	4
2.2 Use Case quality	This area is very large and thus the Use Cases are described at higher level. Thus even the test case planning can be done based on the defined Use Case set, additional work is likely needed to clarify the test case expectations once the test case planning is started. Taking into account the complexity of this requirement specification and its' scope, it was quite natural that the Use Case descriptions are defined at high level. Continuation studies will most likely be started for the most important Use Cases.	3

Table 11 Experiences from Case 4 (cont.)

3. Use Case workshop experiences:		
3.1 Participant availability	In the Use Case workshops, participant availability overall was good. The most important areas were well represented (at least in the beginning of the work).	4
3.2 Workshop efficiency	Efficiency was not very good in the workshops. This may be because lack of preparation, misunderstandings about the workshop purpose or participants may not have understood well enough the scope of the feature itself. It was still possible to create a prioritized Use Case list as a result of the Use Case workshops.	3
3.3 Competence transfer success	In the beginning, the big picture of the existing system level capabilities and product capabilities at the VoLTE geo-redundancy area was very unclear. Therefore one target for the requirement specification was to create overall understanding of the VoLTE geo-redundancy area. This target was achieved. Overall, it is felt that involved person's understanding about the area was significantly increased.	4
4. Testing quality related experiences:		
4.1 Test coverage	Test planning not yet started. Requirement specification work however resulted to work share between the different testing phases and test scope was clarified. Test environment requirements are understood.	Not available
4.2 Testing viewpoint	Test planning not yet started. Use Cases are defined from the customer need viewpoint and thus the customer scope should be well covered in the verification.	Not available